

Universität  
Rostock



Traditio et Innovatio

---

# Neural Text Line Extraction in Historical Documents A Two-Stage Clustering Approach

---

Dissertation

zur

Erlangung des akademischen Grades

doctor rerum naturalium (Dr. rer. nat.)

der Mathematisch-Naturwissenschaftlichen Fakultät

der Universität Rostock

vorgelegt von

Tobias Grüning, geb. am 18.04.1986 in Grevesmühlen

aus Rostock

Rostock, 27.04.2018

[https://doi.org/10.18453/rosdok\\_id00002427](https://doi.org/10.18453/rosdok_id00002427)

*For any search/optimization algorithm, any elevated performance over one class of problems is exactly paid for in performance over another class.*

– no free lunch theorem –

**Gutachter:**

Prof. Dr. Roger Labahn  
Universität Rostock  
Institut für Mathematik

Ph.D. Basilis Gatos  
Institute of Informatics and Telecommunications  
National Center for Scientific Research “Demokritos”

**Jahr der Einreichung:** 2018

**Jahr der Verteidigung:** 2019





# Contents

<b>Notation</b>	<b>III</b>
<b>1. Introduction</b>	<b>1</b>
1.1. Motivation & Thematic Classification . . . . .	1
1.2. Main Contributions . . . . .	3
1.3. Problem Statement . . . . .	5
1.4. Conclusion . . . . .	8
<b>2. Related Work</b>	<b>9</b>
<b>3. Fundamentals</b>	<b>11</b>
3.1. Artificial Neural Networks . . . . .	11
3.1.1. Multilayer Perceptron . . . . .	13
3.1.2. Convolutional Neural Network . . . . .	15
3.1.3. Tasks & Loss Functions for ANNs . . . . .	19
3.1.4. Training of ANNs . . . . .	26
3.1.5. Conclusion . . . . .	30
3.2. Energy Minimization via Graph Cuts . . . . .	31
3.2.1. $\alpha$ - $\beta$ -Swap Energy Minimization . . . . .	31
3.2.2. Optimal $\alpha$ - $\beta$ -Swap via Graph Cuts . . . . .	33
3.2.3. Conclusion . . . . .	37
<b>4. Baseline Detection</b>	<b>39</b>
4.1. Problem Statement . . . . .	40
4.2. Similarity Score . . . . .	41
4.2.1. Motivation & Requirements . . . . .	42
4.2.2. Evaluation Scheme . . . . .	43
4.3. Baseline Detection Method . . . . .	54
4.3.1. Stage I: ARU-Net . . . . .	56
4.3.2. Stage II: Baseline Estimation . . . . .	62
4.4. Experiments . . . . .	75
4.4.1. Influence of Training Sample Number and Data Augmentation	78
4.4.2. Curved and Oriented Text Lines . . . . .	80
4.4.3. U-Net vs. ARU-Net vs. Single-Stage Workflow . . . . .	81
4.4.4. Comparison against the State of the Art . . . . .	82
4.5. Conclusion . . . . .	85

---

<b>5. Baseline to Text Line</b>	<b>87</b>
5.1. Seam Carving . . . . .	87
5.2. Baseline to Text Line Experiments . . . . .	89
5.3. End-to-end Experiments . . . . .	93
5.4. Conclusion . . . . .	95
<b>6. Conclusion &amp; Outlook</b>	<b>97</b>
<b>A. Appendix</b>	<b>99</b>
A.1. Exemplary Results . . . . .	99
<b>List of Figures</b>	<b>113</b>
<b>List of Tables</b>	<b>119</b>
<b>Algorithms</b>	<b>120</b>
<b>Symbols</b>	<b>123</b>
<b>Acronyms</b>	<b>125</b>
<b>Literature</b>	<b>127</b>
<b>Index</b>	<b>141</b>

# Notation

This section gives a concise overview of the notation used throughout this thesis. Concepts which are in our opinion not common are introduced before the first usage. For introduced operations we assume that the operands are of appropriate shape.

## Sets

$\mathbb{A} = \{a, b, c\}$	A set including $a, b, c$
$\mathfrak{A}$	A set (fraktur is used for sets of special importance)
$\mathbb{A} \times \mathbb{B}$	The Cartesian product of sets $\mathbb{A}$ and $\mathbb{B}$
$\mathbb{A}^n$	The set of all $n$ -dim vectors with elements of $\mathbb{A}$
$\mathbb{A}^{n_1 \times n_2}$	The set of all $n_1 \times n_2$ -dim matrices with elements of $\mathbb{A}$
$\mathbb{A}^{n_1 \times \dots \times n_m}$	The set of all $n_1 \times \dots \times n_m$ -dim arrays
$2^{\mathbb{A}}$	The power set of $\mathbb{A}$ (the set of all subsets of $\mathbb{A}$ )
$\mathcal{P}$	A set of sets, e.g., a partition, see Def. 3.2.3

## Convenience

$\emptyset$	The empty set
$\mathbb{N}$	The set of natural numbers ( $\mathbb{N}_0$ including 0)
$\mathbb{R}$	The set of real numbers ( $\mathbb{R}_+$ just positive, including 0)
$\mathbb{Z}, \mathbb{C}$	The set of integers and of complex numbers
$\mathbb{P}[t]$	The set of all polynomials in $t$
$\mathbb{A}^{n \times *}$	The set of all matrices with $n$ rows
$[a, b]$	The real interval including $a$ and $b$
$[n]$	The set of natural numbers from 1 to $n \in \mathbb{N}$

## Numbers, Arrays, and Tuples

$a$	A scalar (real or natural number)
$\boldsymbol{a}$	A vector of scalars
$\boldsymbol{A}$	A matrix of scalars
$\boldsymbol{A}$	An $n$ -dim array of scalars ( $n > 2$ )
$\mathcal{A}$	A tuple (or list or ordered set) of <b>non</b> scalar elements

**Convenience**

$h, i, j, k, l, m, n, w$	Are natural numbers
$a, b$	Are real numbers
$x, y, z$	Could be natural or real numbers
$\mathbf{0}$	The zero vector, matrix, or $n$ -dim array
$e, \pi$	Euler's number, circle number
$(a, b, c)^T$	The vector containing $a, b, c$
$(A, B)$	3-dim array with $A$ and $B$ stacked along the third dim

**Operations on Sets and Tuples**

$A \frown B$	The list $B$ is put at the end of list $A$
--------------	--

**Indexing, Dimensions, and Naming**

For **non** scalars the lower right index position is reserved for indexing purposes. The lower left index position is used to refer to the object's dimensionality. The upper right index position is used (usually in brackets) for the naming of different objects.

$\mathbf{a}_i$	The $i$ -th element of vector $\mathbf{a}$
$A_{i,j}$	The scalar in row $i$ and column $j$ of matrix $A$
$A_{i,:}$	The $i$ -th row of matrix $A$
$A_{:,j}$	The $j$ -th column of matrix $A$
$\mathbf{A}_{i_1, \dots, i_n}$	The scalar with indices $i_1, \dots, i_n$ of $\mathbf{A}$
$\mathbf{A}_{::, i_3, \dots, i_n}$	The matrix for fixed indices $i_3, \dots, i_n$
$\mathcal{A}_i$	The $i$ -th element in the list $\mathcal{A}$
${}_1A$	The number of rows of the matrix $A$
${}_2A$	The number of columns of the matrix $A$
${}_i\mathbf{A}$	The dimensionality of dimension $i$ of $\mathbf{A}$
$\mathbf{a}^{(1)}, \mathbf{a}^{(2)}$	Two different vectors

**Logic Notation**

$\forall$	For all
$\exists$	There exists at least one
$\exists!$	There exists one and only one
$\wedge$	Logical and
$\vee$	Logical or
$\neg$	Logical not

**Convenience**

Let  $A, B \in \{0, 1\}^{n \times m}$  (binary).

$$\begin{array}{ll} A \wedge B & (A \wedge B)_{i,j} = \begin{cases} 1, & \text{if } A_{i,j} = B_{i,j} = 1 \\ 0, & \text{else} \end{cases}, \forall i \in [n], j \in [m] \\ A \vee B & (A \vee B)_{i,j} = \begin{cases} 0, & \text{if } A_{i,j} = B_{i,j} = 0 \\ 1, & \text{else} \end{cases}, \forall i \in [n], j \in [m] \\ \neg A & 0 \text{ becomes } 1 \text{ and vice versa} \end{array}$$

**Matrix Operations**

$A^T$	Transpose of matrix $A$
$A \odot B$	Hadamard product (element-wise multiplication)
$A * B$	Matrix convolution, see Sec. 3.1.3
$A \oplus B$	Morphological dilation ( $A, B$ binary), see [Ser82]
$A \ominus B$	Morphological erosion ( $A, B$ binary), see [Ser82]
$A \circ B$	Morphological opening ( $A, B$ binary), see [Ser82]

**Functions**

$f : \mathbb{A} \rightarrow \mathbb{B}$	Function $f$ mapping from $\mathbb{A}$ to $\mathbb{B}$
$f \circ g$	The composition of $f$ and $g$ ( $f \circ g(x) = f(g(x))$ )
$f(x; \theta)$	Function $f$ in $x$ is parametrized by $\theta$

**Convenience**

$ \cdot $	The absolute value for a scalar argument
$ \cdot $	The number of elements for a <b>non</b> scalar argument
$\lceil a \rceil$	The ceil of scalar $a$
$\lfloor a \rfloor$	The floor of scalar $a$
$\ \cdot\ _2$	The $L_2$ -norm, $\ \mathbf{x}\ _2 = \sqrt{\sum_{i=1}^n \mathbf{x}_i^2}$ , $\mathbf{x} \in \mathbb{R}^n$

**Miscellaneous**

$\mathcal{N}(\mu, \sigma^2)$	Normal distribution with mean $\mu$ and standard deviation $\sigma$
$x \sim \mathcal{N}(\mu, \sigma^2)$	$x$ is sampled from the normal distribution



# 1. Introduction

Accessibility of the valuable cultural heritage of historical documents is an important concern of archives, libraries as well as certain companies, e.g., those specialized in genealogy. After years of digitization at an industrial scale to protect and preserve these valuable goods, millions over millions of scanned pages are stored at servers all over the world [ICH12]. The generic next step is to make the enormous amount of content of these document images accessible and enable humanists, historians, genealogists as well as ordinary people to efficiently work with these documents. Besides the cost- and time-consuming process of manually annotating volumes [CW12], it is subject to current research and scientific discussion how to automate this process [Sán+13]. This thesis investigates a subtask of this information retrieval pipeline, namely the text line extraction.

## 1.1. Motivation & Thematic Classification

Since 2009, tremendous progress in the field of automated text recognition (ATR)<sup>1</sup> [GS08]; [Lei+16] as well as keyword spotting (KWS) [PTV14]; [Str+16a]; [Str+16b] was achieved. The performance of state-of-the-art systems reaches character error rates below 10% for ATR [San+14] and mean average precisions above 0.9 for KWS [Pra+16] for complex handwritten documents. Although efforts are made to develop systems working solely on the rough input image without any a-priori segmentation [Rus+15]; [Blu16]; [KKG16], the best performing recognition systems – with reference to recently hosted competitions – rely on segmented words or text lines as input. Entirely segmentation-free approaches suffer either from an enormous training/inference time and/or, up to now, did not demonstrate its applicability with competitive quality on challenging datasets [Pra+16]. Hence, a workflow which involves a text line extraction followed by the transformation of pixel information into textual information (ATR/KWS) is the widespread standard. This work deals with the first step of the information retrieval pipeline, namely the text line extraction. This is a mandatory step since errors directly effect the performance of the overall information retrieval process. The text line extraction is still unsolved to a certain extend for historical documents due to difficulties like physical degradations (e.g., bleed-through, faded away characters, heterogeneous stroke intensity), image capture conditions (e.g., scan curve, illumination issues), complex layouts

---

<sup>1</sup>ATR = Optical Character Recognition + Handwritten Text Recognition



(e.g., structured documents, marginalia, multi-column layouts, varying font sizes), arbitrary orientations and curved text lines.

The results achieved by state-of-the-art approaches are not satisfying [Zah+07]; [Mur+15], especially if dealing with heterogeneous data. Algorithms are often highly adapted to certain scenarios and need to be re-parametrized by experts; or even fail if applied to collections of different characteristics. Moreover, even state-of-the-art algorithms struggle if they are challenged with historical documents that are characterized by the above mentioned difficulties [Mur+15]. Therefore, this work focuses on the extraction of text lines in arbitrary historical documents.

Traditionally, the problem of text line extraction is tackled by image processing based methods [NG09]; [SSG09]; [Gar+12]; [DKS13]; [AS14]; [RKC14]; [SAE14]. These approaches usually incorporate domain knowledge in form of hand-crafted features. As already mentioned, this process usually necessitates remarkable expert knowledge. Besides these classical methods, deep learning based approaches became omnipresent in the document analysis community within the last years [Vo+17]; [Ten+17]; [Che+17]; [Ren+17]. These methods do not rely on hand-crafted features. Instead, they learn hierarchical representations from raw training data. However, this training data generation is usually cumbersome and expensive. Either plenty of trainings samples are required or the production of single samples is quite elaborate if just a reasonable number of training samples is necessary, e.g., the labeling at pixel level for binarization purposes. The presented work combines deep learning strategies and state-of-the-art image processing based techniques to design a two-stage clustering approach. This is done to overcome the limitations of the single approaches to build a method which constitutes a more powerful and easy to handle framework which is usable by a wide range of people without any remarkable expert knowledge of the fields of computer vision or deep learning.

In this work, the problem of text line extraction is decoupled into the problem of baseline detection and the subsequent problem of the estimation of a polygonal representation of a text line given its detected baseline. This increases the flexibility of the extraction process and allows for an easy adaptation to different ATR/KWS modules. After introducing the problem of baseline detection, a similarity score is introduced to evaluate the quality of baseline detection methods. Finally, we will utilize a newly designed, fully convolutional network (called ARU-Net) to perform a goal oriented pixel labeling. That means the pixels of the input document are assigned to certain classes. Binarization, which is a prior step to many text line extraction algorithms [DKS13]; [RKC14]; [Koo16], is a special form of pixel labeling. But binarization sometimes tends to fail for complex documents and results in hardly correctable errors. Moreover, it just distinguishes between foreground and background, which is usually not sufficient for text line extraction, e.g., ruled lines or graphics have to be separated from text. Finally, binarization algorithms are usually parametrized for a certain collection and necessitate expert knowledge to be adapted to another collection. This motivates the usage of a more sophisticated pixel labeling

method. The proposed ARU-Net can be trained for any pixel labeling task, even to distinguish between different types of text (as we will see later). Furthermore, it is adaptable to different collections and/or scenarios just by retraining. Consequently, there is solely the need for appropriate training data (which will be easy and cheap to produce) instead of expert knowledge to re-parametrize the algorithm (which librarians or digital humanists usually do not have). The ARU-Net output constitutes the input for an image processing based clustering approach. This clustering approach incorporates domain knowledge to improve the overall performance. The approach is based on the state estimation which was introduced in [KC10], firstly used for text line extraction in handwritten documents in [KC12], and further extended in [Grü+17]; [Ahn+17]. The latter was referred to as current state of the art in a recent review [EGO17], because it won the last two (to that point) competitions on text line extraction organized in conjunction with the ICDAR conference. However, our proposed two-stage baseline detection method proves its superiority over the previous state of the art. It substantially outperforms the submissions to 3 recently organized competitions on origin point/baseline detection and is already usable (and used) by a wide audience via the Transkribus<sup>2</sup> platform.

After detecting the baselines, we utilize an approach which is based on dynamic programming to determine polygonal representations for the text lines. In the end, the influence on the overall information retrieval pipeline of the proposed text line extraction method is investigated. Fig. 1.1 illustrates a snippet of the cBAD test set [Die+17b] with the detected baselines and the estimated polygonal chains representing the text lines. This snippet will cross our path several times throughout this thesis (until we achieve the depicted results).

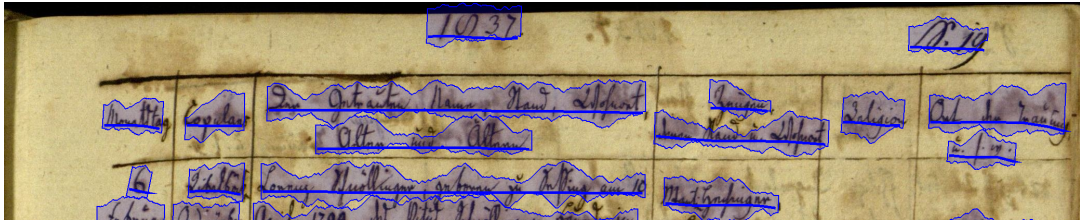


Figure 1.1.: **Baselines and text lines for an image snippet of [Die+17b]** – Illustration of the detected baselines (solid blue lines) and the extracted text lines (blue underlay) for a snippet of a historical document image which were estimated by the proposed text line extraction method.

## 1.2. Main Contributions

Here, we will briefly list the main contributions of this thesis followed by an outline of this thesis. Main contributions:

<sup>2</sup><https://transkribus.eu>

- Formulation of the problem of text line extraction in a formal way, especially a definition of a correct text line is given in a practically motivated way in form of the text line intuition.
- Introduction of the decoupling of the stated problem in a baseline detection problem and a text line extraction problem.
- Development of a similarity score for the baseline detection problem.
- Introduction of a newly designed deep neural network (ARU-Net) for pixel labeling along with a meaningful parametrization (which is the result of an extensive search in the hyperparameter space and leads to impressive results over a wide range of different scenarios) – the ARU-Net and its training framework are open source<sup>3</sup>.
- Introduction of the new concept of learned separators to handle complex layouts instead of an a-priori page segmentation or white-/blackrun calculation.
- Introduction of a two-stage workflow which combines the developed ARU-Net with state-of-the-art image processing techniques. The resulting method outperforms the previous state of the art on several data sets.
- The developed method was tested for a variety of different data sets, especially, the influence of the amount of training data as well as data augmentation strategies were investigated to demonstrate the applicability of the proposed approach at an industrial scale.
- Introduction of a dynamic programming based method which utilizes the calculated baselines to estimate polygonal chain representations of the text lines.
- Evaluation of the entire information retrieval pipeline in an end-to-end fashion.
- The entire text line extraction method was made available via the Transkribus platform and is used by a wide range of researchers from different domains.

The remainder of the thesis is structured as follows: After introducing the text line extraction problem in Sec. 1.3, Chap. 2 briefly introduces some related work. In Chap. 3 the fundamentals for the baseline detection method are given. After thoroughly introducing the fully convolutional network, an energy minimization framework based on graph cuts is introduced. Chap. 4 focuses on the baseline detection and constitutes the main chapter of this thesis. Besides the motivation of the baseline detection as a first step of the text line extraction framework, the developed similarity score to evaluate the performance of a baseline extraction method is described. The remainder of this chapter is dedicated to the developed baseline detection method. After introducing the method, its capabilities are demonstrated in various experiments. In Chap. 5 a method to estimate polygonal chains given the detected baselines is described. This method is evaluated on a recently hosted text line extraction competition with respect to the intersection-over-union measure. Afterwards, the end-to-end performance of the entire system is investigated. For this purpose, a system for automated text recognition is trained and the transcriptions of the entire system are evaluated by means of the character error rate. Finally, Chap. 6 concludes the thesis.

---

<sup>3</sup><https://github.com/TobiasGruening/ARU-Net>

## 1.3. Problem Statement

This section introduces definitions of basic terms to state the text line extraction problem in a formal way. For instance, image is a term frequently used in the literature, nevertheless it is related to various different objects. Hence, it is necessary here to clarify exactly what is meant by image.

**Definition 1.3.1** (image, pixel, intensity, coordinate). A matrix  $I \in [0, 1]^{h \times w}$  is called (*gray-scale*) *image* of height  $h \in \mathbb{N}$  and width  $w \in \mathbb{N}$ . An element  $\mathbf{p} = (i, j) \in [h] \times [w]$  is called *pixel* of  $I$ . The matrix value  $I_{\mathbf{p}} = I_{i,j}$  of row  $i$  and column  $j$  is called *intensity* of pixel  $\mathbf{p}$ .  $y = \mathbf{p}_1$  and  $x = \mathbf{p}_2$  are called *coordinates* ( $y$ - and  $x$ -coordinate). An image of zeros and ones  $I \in \{0, 1\}^{h \times w}$  is called *binary image*.

**Remark 1.3.2.** It is common in computer vision that the first coordinate constitutes the  $y$ -coordinate and the second one the  $x$ -coordinate. Furthermore, the  $y$ -axis is typically inverted compared to the Cartesian coordinate system. We follow this convention throughout this thesis.

**Remark 1.3.3.** Pixel coordinates are natural numbers. Nevertheless, we assume that all operations which are defined in  $\mathbb{R}^2$  are also applicable on pixels. A pixel is mapped onto  $\mathbb{R}^2$  in the generic way. Any  $\mathbf{x} \in \mathbb{R}_+ \times \mathbb{R}_+$  is mapped back into the pixel space by  $\mathbf{p} = (\text{round}(\mathbf{x}_1), \text{round}(\mathbf{x}_2))^T$  with  $\text{round} : \mathbb{R}_+ \rightarrow \mathbb{N}$  defined as

$$\text{round}(x) := \begin{cases} \lfloor x \rfloor & , \text{ if } x > 1 \wedge x - \lfloor x \rfloor \leq 0.5 \\ \lceil x \rceil & , \text{ else} \end{cases}.$$

Note that  $\text{round}(0.1) = 1$  because a pixel coordinate has to be a natural number. Image means gray-scale image for the rest of this work.  $_1I$  denotes the height of image  $I$ ,  $_2I$  denotes the width, analogously. For visualization purposes a pixel intensity value of 1 means white and 0 means black.

**Definition 1.3.4** (image space). The set of all possible images

$$\mathfrak{I} := \bigcup_{h,w \in \mathbb{N}} [0, 1]^{h \times w}$$

is called *image space* ( $\mathfrak{I}$ ).

**Remark 1.3.5.** A *colored image* can be regarded as three stacked gray-scale images, each encoding one of the following colors – red ( $R \in \mathfrak{I}$ ), green ( $G \in \mathfrak{I}$ ), blue ( $B \in \mathfrak{I}$ ). In this work, we follow the recommendation ITU-R BT.601-7 [Ser11] for the conversion of a colored image to a gray-scale image

$$I = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B.$$

If the colored image is available, we usually use it for visualization even though it is converted to its gray-scale version for calculations.

Everybody has an intuition about what a text line is. However, this intuition usually differs between people and even between scenarios for the same person. Thus, it is impossible to state a formal definition which holds in general (for all intuitions) and covers all characteristics to clearly describe a text line. Therefore, a text line is introduced in a quite general way.

**Definition 1.3.6** (text line, text line space). Given an image  $I \in \mathfrak{I}$  a *text line* is a subset of the image pixels

$$\mathbb{P} \subset [{}_1I] \times [{}_2I].$$

The set of all possible text lines

$$\mathfrak{P}_{TL} := 2^{\mathbb{N} \times \mathbb{N}}$$

is called *text line space* ( $\mathfrak{P}_{TL}$ ).

Hence, a text line could be anything. As stated above there is no closed definition to describe how a "correct" text line looks like. To overcome this issue, we introduce the text line intuition as a function which maps any image to a set of text lines.

**Definition 1.3.7** (text line intuition, text line ground truth). A function  $\iota_I : \mathfrak{I} \rightarrow 2^{\mathfrak{P}_{TL}}$  is called *text line intuition (TLI)* if and only if (iff) it reflects the intuition about text lines of an (human) operator. The set of text lines  $\iota_I(I)$  is referred to as (*text line*) *ground truth (GT)* for  $I \in \mathfrak{I}$ .

A TLI is given by an (human) operator and usually motivated by an application. Basically, different TLIs differ in the understanding of what a text line is and how it should be represented. The major differences of practically relevant TLIs are described and visualized in the introduction of Chap. 4. However, since there are no well-defined computation rules for a TLI, the realization for a certain image has to be done manually by the (human) operator. This takes quite a long time and is not suitable for large scale applications. Hence, it is within the scope of this work to eliminate the operator-part in this process. The text line extraction should be automated. Thus, there has to be a purely automatic method which produces text line hypotheses for an image. The calculation rules of this function have to be known. We call such a function text line extractor.

**Definition 1.3.8** (text line extractor, text line hypothesis). A function  $\iota_E : \mathfrak{I} \rightarrow 2^{\mathfrak{P}_{TL}}$  is called *text line extractor (TLE)* iff its calculation rules are well-defined. The set of text lines  $\iota_E(I)$  is referred to as (*text line*) *hypothesis (HYP)* for  $I \in \mathfrak{I}$ .

For the rest of this thesis, we aim for the design of a TLE which "optimally" approximates a given TLI. To evaluate the quality of a TLE for a certain image a similarity score is introduced. A similarity score should (as the name implies) assign a value to two sets of text lines which represents their similarity.

**Definition 1.3.9** (similarity score). A function  $\langle \cdot, \cdot \rangle_{TL} : 2^{\mathfrak{P}^{TL}} \times 2^{\mathfrak{P}^{TL}} \rightarrow [0, 1]$  is called *similarity score*.

Some application oriented similarity scores are introduced in Chap. 5. Basically, a similarity score assigns high values to sets of text lines if they are very similar. Thus, a value close to 1 ( $\langle \iota_I(I), \iota_E(I) \rangle_{TL} \approx 1$ ) indicates a high accuracy of the TLE  $\iota_E(I)$  for the image  $I$  given a TLI  $\iota_I$ . Accordingly, a TLE is optimal for a TLI and a similarity score  $\langle \cdot, \cdot \rangle_{TL}$  iff it solves

$$\iota_E^* = \arg \max_{\iota_E \text{ is TLE}} \sum_{I \in \mathfrak{I}} \langle \iota_I(I), \iota_E(I) \rangle_{TL}. \quad (1.3.1)$$

As already mentioned, there are no closed computation rules for a certain TLI. Hence, the optimization problem defined in Eq. (1.3.1) is not manageable due to the infinite number of possible images. Therefore, the (human) operator has to choose a representative subset of  $\mathfrak{I}$  which (ideally) is sampled independent and identically distributed (i.i.d.) from all images of interest. The combination of this subset of images and their corresponding realizations of the TLI (GT) is called *test set* ( $\mathfrak{T}$ )

$$\mathfrak{T}_{\iota_I} = \left\{ \left( I^{(i)}, \iota_I(I^{(i)}) \right) \mid i \in [n_{test}], I^{(i)} \in \mathfrak{I} \right\}, \quad (1.3.2)$$

where  $n_{test} \in \mathbb{N}$  is the number of test samples. The test set should represent the TLI (on a meaningful subset of  $\mathfrak{I}$ ) and is used to evaluate the quality of a TLE. For each test set there is always a fixed TLI, this is denoted by the respective subscript. The production of  $\mathfrak{T}_{\iota_I}$  has to be done by the (human) operator and is, even for a subset of reasonable size, time-consuming and expensive. Finally, the quality of a TLE is measured by means of the similarity score on the test set.

**Definition 1.3.10** (test score). Assume there is a test set  $\mathfrak{T}_{\iota_I}$  and a similarity score  $\langle \cdot, \cdot \rangle_{TL}$ , the *test score* ( $\tau_{test}$ ) assigns a score to each TLE and is defined as

$$\tau_{test}(\iota_E) := \frac{1}{|\mathfrak{T}_{\iota_I}|} \sum_{(I, \iota_I(I)) \in \mathfrak{T}_{\iota_I}} \langle \iota_I(I), \iota_E(I) \rangle_{TL}.$$

Taking into account the test set during the design phase of the TLE is fraught with risk. In the worst case the TLE is just memorizing  $\mathfrak{T}_{\iota_I}$  and it does not approximate the underlying TLI at all. This behavior is referred to as *overfitting*. The performance of a system which overfits on a certain set is typically very poor on arbitrary inputs which are not part of this set.

To overcome this issue a *train set* ( $\mathfrak{T}'$ )

$$\mathfrak{T}'_{\iota_I} = \left\{ \left( I^{(i)}, \iota_I(I^{(i)}) \right) \mid i \in [n_{tr}], I^{(i)} \in \mathfrak{I} \right\}. \quad (1.3.3)$$

is introduced. It is set up like the test set, especially the same TLI is used to generate both sets. But the sets can differ in their sizes and have to differ in the images used,

i.e.,  $\mathfrak{T}'_{\iota_I} \cap \mathfrak{T}_{\iota_I} = \emptyset$ .

A TLE which was designed using solely  $\mathfrak{T}'_{\iota_I}$  and which performs well on  $\mathfrak{T}_{\iota_I}$  is much more likely approximating the underlying TLI than a system which was designed using both sets. Thus, the major difficulty is to design a system which generalizes well from the train set to the test set. The problem investigated in this thesis can now be stated as follows.

**Text Line Extraction Problem** *Assume there is a text line intuition along with a test set  $\mathfrak{T}_{\iota_I}$  and a train set  $\mathfrak{T}'_{\iota_I}$  which were generated by means of the TLI. Design a text line extractor ( $\iota_E^*$ ) which utilizes just  $\mathfrak{T}'_{\iota_I}$  and maximizes the test score  $\tau_{test}$  on  $\mathfrak{T}_{\iota_I}$ .*

Moreover, we do not just aim for the design of a TLE for a certain TLI but for a design which is adaptable to different TLIs. Hence, the designed TLE should be of the form

$$\iota_E(\cdot; \theta) : \mathfrak{I} \rightarrow 2^{\mathfrak{P}_{TL}}$$

with a vector of parameters  $\theta \in \mathbb{R}^N$ . Given a new TLI, an adaptation of  $\theta$  should be sufficient to solve the above stated problem. Furthermore, the designed TLE should allow for an "easy" estimation of  $\theta$ . That is to say, for a certain TLI it should be possible to estimate  $\theta$  without any expert knowledge and at a reasonable cost. This is a mandatory characteristic for a TLE which should be used at an industrial scale and is therefore one major requirement within this thesis.

## 1.4. Conclusion

In this chapter, we have motivated the topic of this thesis. The necessity of a robust and flexible text line extraction method in the information retrieval pipeline for scanned documents was mentioned. Especially, there is a lack of methods which are adaptable to entirely new scenarios without any notable expert knowledge. Afterwards, the main contributions of this thesis were stated. Finally, the problem of text line extraction was stated in a formal way.

## 2. Related Work

A myriad of solutions to the text line extraction problem has been proposed in the literature. Likforman-Sulem et al. [Zah+07] and Eskenazi et al. [EGO17] give surveys which cover the last two decades. Especially, the recently published review [EGO17] provides a comprehensive overview of methods of the last ten years. Because of that, we will not provide a global summary of methods and limit ourselves towards a brief introduction of approaches which are relevant for this work. We will evaluate each method (or class of methods) with respect to the requirements for a text line extractor which were stated in the previous section.

In [AS14]; [NG09]; [SAE14], the principle of dynamic programming is utilized to calculate cost optimal paths passing the image from left to right to separate different text lines from each other. Basically, the above cited methods differ in the way the images are pre-processed and in the formulation of the cost function which is optimized. The major drawback of these methods is that they solely work for a subset of all images of interest. I.e., they are just applicable for images with (roughly) horizontally oriented text lines which range from the left to the right image border. In the words of [EGO17], these methods are *algorithm constrained*. That means, they are limited to a pre-defined scenario and not adaptable to different TLIs. Nevertheless, these methods do not necessarily rely on a binarization and work very well in their pre-defined constrained environment. Therefore, they are the basis for our baseline-to-text-line method which will be described in Chap. 5.

Ryu et al. [RKC14] propose an algorithm which uses characteristics (so-called states) of extracted connected components to assign costs to certain clustering results. These states encode local text orientation and interline distances (whose meaning is intuitively clear but will be introduced later in this work). Subsequently using four different operations (merge, split, merge-split, merge-merge-split) on an initial coarse clustering, the costs are minimized to obtain an optimal clustering, which leads to the final text line segmentation. Ahn et al. [Ahn+17] improve this approach by the introduction of a newly developed binarization method and an improved clustering process. Grüning et al. [Grü+17] extended the approach of Ryu et al. so that it is applicable for more general superpixels which do not rely on a binarization. Furthermore, they introduce a new clustering procedure which does not rely on a coarse initial clustering. These algorithms belong to the class of *parameter constrained* (cf. [EGO17]) methods. Consequently, they are more adjustable to different TLIs compared to the algorithm constrained methods. The required parameter tuning typically necessitates expert knowledge which is not acceptable for



applications which should be used at industrial scale. Nevertheless, these methods constitute an important ingredient for our proposed baseline detection methodology. The ideas of these methods allow for an easy incorporation of domain knowledge, e.g., of basic characteristics of text lines as well as of provided layout information, into our proposed text line extraction framework, cf. Sec. 4.3.2.

Besides these “classical” approaches, which are based on image processing techniques, methods based on machine learning gained importance within the last two years. Eskenazi et al. [EGO17] classify the neural network based approaches as *potentially unconstrained* methods. This is due to the fact that the capability of neural network based methods is mainly determined by the given training data. Hence, they can potentially adapt to various TLIs. Moysset et al. [Moy+15] propose a method based on a recurrent neural network. The network is trained given only the number of lines in the image utilizing Connectionist Temporal Classification which was introduced to train networks for handwriting text recognition and allows for ground truth data without any alignment. The trained neural network predicts confidences for the vertical coordinates of the image to belong either to the classes line or interline. Further post-processing of the neural network output is performed to detect the text lines. In follow-up works, they formulated the problem as a regression problem [Moy+17]; [MKW17]. The recurrent neural network directly predicts bounding boxes as well as the start of each text line, respectively. Besides this regression based approach, classification based approaches were proposed most recently. In contrast to the approach of Moysset et al., these methods perform a pixel labeling to classify each image pixel (instead of classifying rows of pixels, only). For instance, Renton et al. [Ren+17] propose a fully convolutional network (FCN) based on dilated (or atrous) convolutions to classify pixels as text line main body or not. The classification results are utilized to extract the text line information. These techniques are currently very popular, e.g., four of the five participants of the cBAD: ICDAR2017 Competition on Baseline Detection [Die+17a] use methods relying on FCNs. The method introduced in this work relies (in the first stage) on such a neural network based pixel labeling, see Sec. 4.3.1. In contrast to the above mentioned methods which (to a greater or lesser extent) directly use the neural network output as result, the proposed method utilizes the neural network output as input for a subsequent sophisticated clustering approach, see Sec. 4.3.2.

## 3. Fundamentals

In this chapter, we introduce the fundamentals for the techniques used to work on the text line extraction problem as stated in Sec. 1.3. Each technique is thoroughly introduced in a mathematical fashion. Several references for further studies beyond this thesis are given. After introducing a technique a short conclusion is given. The conclusion states the main points of the technique and motivates its usage to solve the text line extraction problem. The introduced techniques are:

- (1) Artificial Neural Networks
- (2) Energy Minimization via Graph Cuts

### 3.1. Artificial Neural Networks

The question whether programmable computers could become intelligent dates back to a time 100 years before the first one was build [ML43]. Nevertheless, it took quite a while until the first steps towards *artificial intelligence (AI)* were made. The first success stories of AI were limited to problems with well-defined, sterile environments. For this kind of problems, it was possible to formulate a list of formal mathematical equations. As a result, the design of hard-coded strategies constituted the solution to this problems. Hence, the first AI systems were based on a static set of rules. One of the most prominent systems among the knowledge-based systems is IBM's Deep Blue chess-playing system which defeated the world champion Garry Kasparov in 1997 [Hsu02].

However, a major problem of this kind of knowledge-based systems is that they struggle if confronted with problems which suffer from a complex or intractable problem formulation. Therefore, powerful AI systems should be able to acquire their own knowledge by extracting patterns from raw data. This capability is known as *machine learning (ML)*. The performance of machine learning methods strongly rely on the representation of the data they are trained on. E.g., to determine whether there is a face in an image or not is quite simple if one gets number and position of eyes, noses, mouths, ears, ... instead of the raw pixel intensities. The design of meaningful high level *features* like the above-mentioned could be done by (human) experts. The so-called handcrafted features are used in a large variety of different problems. Nevertheless, there are plenty of problems for which the formulation of a meaningful set of features (and their detection) is nearly as challenging as the

original problem. Especially for computer vision tasks this is true. The detection of eyes, noses, mouths, ears, ... is nearly as complex as the detection of faces. In *deep learning (DL)* this problem is tackled by an automatic design of high-level features that are expressed in terms of lower level (simpler) features. No (human) expert is required in this design phase.

The past decade has seen the rapid development of DL [Sch15]; [LBH15]. Researchers worked on and solved problems in different domains in a quality which results in a gain of attention even in the public at large. Especially large IT-companies like Google, Amazon, Apple, Microsoft foster the DL research and incorporate newly developed techniques directly in their products. Personal assistants, e.g., Google Home, Amazon Echo, Siri, and Cortana, are of special importance regarding the publicity. These products are a direct consequence of advances in speech recognition [D+10]; [Dah+12]; [Hin+12]; [GMH13] and natural language processing [Col+11]; [Kum+16]. There are various other domains which benefit from recent progress, e.g., automation (self-driving cars [Far+13]), automatic image captioning [KSZ14a]; [KSZ14b]; [Mao+14]; [Fan+15]; [Vin+15]; [Xu+15], neural machine translation [SVL14]; [BCB14], information extraction in biomedical images [RFB15], GO-computer-programs which are able to beat the worlds best human players [Sil+16]; [Sil+17], to name but a few.

The omnipresent technique and basis for the above-mentioned success stories is the *artificial neural network (ANN)*. This approach, as its name implies, tries to understand and imitate the human brain. The term "(artificial) neural network" covers a wide range of different models which aim to describe neural behavior in biological systems in a mathematically fashion [MP43]; [Heb49]; [WH60]; [Ros61]; [RHW86]. Nowadays, there are plenty of different ANN models with various properties to distinguish between them. One major difference is the way the model is internally connected. There are models whose connections form cycles, so-called *recurrent neural networks*. These networks are widely used for sequence processing problems, e.g., speech recognition [GMH13], language processing [Kum+16], language modeling [Mik+10]; [Mik+11]; [MZ12], automatic translation [SVL14], and automatic text recognition [Gra+09]; [Lei+16]; [Str+16a]. The other category, the *feedforward neural networks*, has acyclic connections. In this work, we will limit ourselves towards the feedforward neural networks because they constitute the most popular ANN models for computer vision tasks [KSH12]; [Far+13]; [Goo+13]; [Gir+16]. Next, we will introduce the basic concepts of modern feedforward neural networks which will be used in this thesis. Based on the classical multilayer perceptron the convolutional neural network is introduced. This type of ANN is the basis for the method described in Chap. 4.

### 3.1.1. Multilayer Perceptron

In the 1950s, [Ros58]; [Ros61] introduced the *perceptron*. This was the first model with trainable weights and is widely seen as the basis for modern ANNs. In the late 1980s, the first deep neural network architectures were introduced by stacking multiple perceptrons to build the so-called *multilayer perceptron (MLP)* [RHW86]; [Wer88]. Notably, these stacked perceptrons were extended versions of the originally proposed perceptron (see [RHW86]). In the following, we will introduce the multilayer perceptron in our own notation which is close to the standard notation and is used for the rest of the work. First, we will introduce all necessary components to set up the MLP.

**Definition 3.1.1** (logit function). For a given matrix  $W \in \mathbb{R}^{n \times m}$  and a vector  $\mathbf{b} \in \mathbb{R}^m$  the function  $\Phi^{W,\mathbf{b}} : \mathbb{R}^n \rightarrow \mathbb{R}^m$  defined as

$$\Phi^{W,\mathbf{b}}(\mathbf{x}) := W^T \mathbf{x} + \mathbf{b}$$

is called *logit function*. The resulting vector  $\tilde{\mathbf{x}} = \Phi^{W,\mathbf{b}}(\mathbf{x})$  is called *logit*.

The free model parameters  $W$  and  $\mathbf{b}$  are called *weights* and *bias*, respectively. They determine the behavior of the logit function. To incorporate the capability to model non-linear dependencies between input and output an activation function is introduced.

**Definition 3.1.2** (activation function). A continuous and monotonically increasing function  $\psi : \mathbb{R} \rightarrow \mathbb{R}$  is called *activation function*.

Remarkably, in the literature there are also non-monotone activation functions, e.g., radial basis function [BL88]; [LB88]; [PS91], but these are not of interest within this work.

**Remark 3.1.3.** For non-scalar inputs the activation function is applied element-wise for its input. For instance, for a given vector  $\mathbf{x} \in \mathbb{R}^n$

$$\psi(\mathbf{x}) = (\psi(\mathbf{x}_1), \dots, \psi(\mathbf{x}_n))^T$$

holds.

There are various different commonly used activation functions. The tanh, logistic, and relu functions (f.l.t.r)

$$\psi(x) = \tanh(x), \quad \psi(x) = \frac{1}{1 + e^{-x}}, \quad \psi(x) = \max\{0, x\}$$

are among the most frequently used ones in state-of-the-art systems. Besides these activation functions which are applied element-wise to their inputs, there are activation functions which take into account the entire input. The so-called softmax activation is of importance especially for classification purposes.

**Definition 3.1.4** (softmax). The function  $\psi : \mathbb{R}^n \rightarrow (0, 1]^n$  defined as

$$\psi(\mathbf{x}) := \left( \frac{e^{x_1}}{\sum_{i=1}^n e^{x_i}}, \dots, \frac{e^{x_n}}{\sum_{i=1}^n e^{x_i}} \right)^T$$

is called *softmax*.

The softmax activation outputs are all positive and sum up to one. Hence, they can model a (probability) distribution and are often used to activate the output layer of a neural network which is used for classification purposes.

**Remark 3.1.5.** In the literature, the term logit is often solely used for the input to the softmax activation. We use it for the input to any activation function.

The composition of logit function and activation function is called layer.

**Definition 3.1.6** (layer). The pair  $(\Phi^{W,b}, \psi)$  of logit function and activation function is called *layer*. For a given *input*  $\mathbf{x} \in \mathbb{R}^n$ , the layer's *output*  $\mathbf{y} \in \mathbb{R}^m$  is calculated as follows

$$\mathbf{y} = \psi \circ \Phi^{W,b}(\mathbf{x}) = \psi(\Phi^{W,b}(\mathbf{x})).$$

The components  $\mathbf{y}_i$ ,  $i \in [m]$  of the output vector are called *units*.

For the reason of simplicity, we usually write  $\Phi^{W,b,\psi}$  for a layer and  $\Phi^{W,b,\psi}(\mathbf{x}) = \psi(\Phi^{W,b}(\mathbf{x}))$  holds. Several layers could be stacked in a way that a layer's output constitutes the next layer's input. One has to ensure that the output dimension of a layer equals the input dimension of the next layer. This has to be done by choosing the layers weight and bias dimensions correctly. The resulting model is referred to as multilayer perceptron.

**Definition 3.1.7** (multilayer perceptron). For  $z_{in}, z_{out}, l \in \mathbb{N}$  with  $l > 1$ , we call a set of layers  $\Phi^{W^{(1)},b^{(1)},\psi_1}, \dots, \Phi^{W^{(l)},b^{(l)},\psi_l}$  *multilayer perceptron (MLP)* iff

$${}_1W^{(1)} = z_{in} \wedge {}_2W^{(l)} = z_{out} \wedge \forall i \in [l-1] : {}_2W^{(i)} = {}_1W^{(i+1)}. \quad (3.1.1)$$

For a given *input*  $\mathbf{x} \in \mathbb{R}^{z_{in}}$  the MLP's *output*  $\mathbf{y} \in \mathbb{R}^{z_{out}}$  is calculated by subsequently updating the layers

$$\mathbf{y} = \Phi^{W^{(l)},b^{(l)},\psi_l} \circ \dots \circ \Phi^{W^{(1)},b^{(1)},\psi_1}(\mathbf{x}).$$

The last layer  $\Phi^{W^{(l)},b^{(l)},\psi_l}$  is referred to as *output layer*. The propagation of the input through the layers is called *forward propagation*.

The input/output (I/O) dimensions, the number of layers  $l$ , the dimensionality of the weights and biases as well as the choice of the activation functions are called

*hyper parameters* of the MLP. The weights and biases themselves constitute the *model parameters*.

**Remark 3.1.8.** An MLP is a function mapping an input vector to an output vector  $\Phi(\cdot; \theta) : \mathbb{R}^{z_{in}} \rightarrow \mathbb{R}^{z_{out}}$ ,  $\theta$  constitutes the model parameters.

Since the behavior of the function is parameterised by the model parameters  $\theta$ , an MLP is capable of instantiating many different functions. A commonly used example to demonstrate the representative power of MLPs is the so-called XOR problem. The XOR function is not linearly separable but it is easy to see that a 2-layer MLP can approximate the XOR function, see [GBC16] [pp. 171-177].

### 3.1.2. Convolutional Neural Network

In computer vision, the task is typically an approximation of a function generating some kind of output given an image (cf. Def. 1.3.1) as input. To tackle this kind of problem one could flatten the input image to get a vector and use an MLP. However, there are (at least) three major drawbacks of the classical MLP used for visual inputs which motivate an extension of it.

First, the number of model parameters is intractably large for real world problems. For instance, a layer taking an  $2000 \times 2000$  image as input and producing an output of dimension 4000 has  $1.6 \cdot 10^{10} + 4000$  free parameters. Second, it could be useful to detect the same features at different locations of the image. In other words, equivariance to translation is desired. Finally, for an MLP the input dimension and output dimension are fixed. Of course, flexibility concerning these dimensions is also a desired property if working on computer vision problems.

The *convolutional neural network (CNN)* which was introduced by LeCun et al. [LeC+89]; [LeC+90]; [LeC+98] overcomes this limitations. The CNN differs from the MLP mainly in the input/output behavior and in the way the logit function is defined. First, we will define the convolution logit function which is the basis for the CNN. The convolution logit function is based on the discrete convolution.

**Definition 3.1.9** (discrete convolution). Let  $\mathbb{D} = \mathbb{Z}^n$  be the domain of the complex valued functions  $f, g : \mathbb{D} \rightarrow \mathbb{C}$ , the operation

$$(f * g)(d) := \sum_{t \in \mathbb{D}} f(t)g(d - t)$$

is called *discrete convolution*.

To simplify the notation which is necessary to introduce the convolution logit function, the convolution for two matrices is defined based on the discrete convolution.

**Definition 3.1.10** (matrix convolution). For  $K \in \mathbb{R}^{k_h \times k_w}$  and  $X \in \mathbb{R}^{h \times w}$  let

$$h_l = \begin{cases} \lfloor k_h/2 \rfloor & , \text{ if } k_h \text{ is odd} \\ k_h/2 - 1 & , \text{ else} \end{cases}, \quad w_l = \begin{cases} \lfloor k_w/2 \rfloor & , \text{ if } k_w \text{ is odd} \\ k_w/2 - 1 & , \text{ else} \end{cases}.$$

The operation  $K * X \in \mathbb{R}^{h \times w}$  defined as

$$(K * X)_{i,j} := \sum_{l=-h_l}^{\lfloor k_h/2 \rfloor} \sum_{m=-w_l}^{\lfloor k_w/2 \rfloor} K_{l+h_l+1, m+w_l+1} X_{i+l, j+m} \quad \forall i \in [h], j \in [w] \quad (3.1.2)$$

with  $X_{i,j} = 0$  for  $i \notin [h] \vee j \notin [w]$  is called *matrix convolution*.

The matrix convolution can be reduced to the discrete convolution in the following way. Think of  $X$  and  $K$  as two functions with discrete domains

$$\begin{aligned} f_X : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{R} \quad \text{by} \quad f_X(i, j) &:= \begin{cases} X_{i,j} & , \text{ if } i \in [h] \wedge j \in [w] \\ 0 & , \text{ else} \end{cases}, \\ f_K : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{R} \quad \text{by} \quad f_K(i, j) &:= \begin{cases} K_{i,j} & , \text{ if } i \in [k_h] \wedge j \in [k_w] \\ 0 & , \text{ else} \end{cases}. \end{aligned}$$

The function  $\hat{f}_K(i, j) = f_K \circ \tau(i, j)$  results from  $f_K$  by translating its support such that it is centered at the origin. The translation function  $\tau : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z} \times \mathbb{Z}$  is defined as  $\tau(i, j) := (i + h_l + 1, j + w_l + 1)$  with  $h_l, w_l$  of Def. 3.1.10. Furthermore, let  $\tilde{f}_K(i, j) := \hat{f}_K(-i, -j)$ . If the result of the discrete convolution of  $\tilde{f}_K * f_X$  is reduced to the spatial domain of  $X$  ( $[h] \times [w]$ ) and written as a matrix, one gets Eq. (3.1.2).

**Remark 3.1.11.** The matrix convolution is often parametrized by natural numbers called *strides*. The strides determine the reduction in the height and width of  $X$ . For given strides  $s_y, s_x$  the matrix convolution is defined as

$$(K * X)_{i,j} := \sum_{l=-h_l}^{\lfloor k_h/2 \rfloor} \sum_{m=-w_l}^{\lfloor k_w/2 \rfloor} K_{l+h_l+1, m+w_l+1} X_{i \cdot s_h + l, j \cdot s_w + m} \quad \forall i \in [\lceil h/s_h \rceil], j \in [\lceil w/s_w \rceil]. \quad (3.1.3)$$

Hence, the standard matrix convolution has strides  $s_h = s_w = 1$ .

In the following, the inputs and outputs are usually 3-dim arrays. An image as in Def. 1.3.1 can also be written as a 3-dim array ( $I \in [0, 1]^{w \times h}$  becomes  $\mathbf{I} \in [0, 1]^{w \times h \times 1}$ ). On this basis, we call the first two dimensions of 3-dim arrays *spatial dimensions*. The first dimension is usually referred to as *height* and the second dimension is referred to as *width*. The third dimension is referred to as (*representative*) *depth*. E.g., a grey-scale image has a representative depth of one, a colored image has a depth of three.

The matrix convolution could be applied on several different matrices and the results could be stacked. As a result, this generic extension of the matrix convolution to 3-dim arrays along with the introduction of a bias term constitute the convolutional logit function.

**Definition 3.1.12** (convolutional logit function). For a 4-dim array called *kernel*  $\mathbf{K} \in \mathbb{R}^{k_h \times k_w \times z_{in} \times z_{out}}$  and a bias vector  $\mathbf{b} \in \mathbb{R}^{z_{out}}$ , the function  $\Phi_{conv}^{\mathbf{K}, \mathbf{b}} : \mathbb{R}^{* \times * \times z_{in}} \rightarrow \mathbb{R}^{* \times * \times z_{out}}$  defined as

$$\Phi_{conv}^{\mathbf{K}, \mathbf{b}}(\mathbf{X})_{::, l} := \sum_{k=1}^{z_{in}} \mathbf{K}_{::, k, l} * \mathbf{X}_{::, k} + \mathbf{b}_l \quad \forall l \in [z_{out}]$$

is called *convolutional logit function*.  $\Phi_{conv}^{\mathbf{K}, \mathbf{b}}(\mathbf{X})$  is called *convolutional logit*.

As a reminder, the dimensionality "star" represents an arbitrary dimension. For the addition of the bias term (which is a scalar) and the result of the convolution (which is a matrix) the bias term is broadcasted to the matrix dimension. We will extend the softmax activation of Def. 3.1.4 to be capable to process a convolutional logit.

**Definition 3.1.13** (convolutional softmax). The function  $\psi : \mathbb{R}^{* \times * \times z} \rightarrow (0, 1]^{* \times * \times z}$  defined as

$$\psi(\mathbf{X})_{i, j, :} := \left( \frac{e^{\mathbf{X}_{i, j, 1}}}{\sum_{k=1}^z e^{\mathbf{X}_{i, j, k}}}, \dots, \frac{e^{\mathbf{X}_{i, j, z}}}{\sum_{k=1}^z e^{\mathbf{X}_{i, j, k}}} \right)^T \quad \forall i \in {}_1\mathbf{X}, j \in {}_2\mathbf{X}$$

is called *convolutional softmax*.

The outputs of the convolutional softmax are all positive. Furthermore, they sum to one for each spatial position (summation over the representative depth). Hence, the convolution softmax can model position dependent (probability) distributions. Finally, the combination of the convolutional logit function and an activation function (either of Def. 3.1.2 or the convolutional softmax) yields the convolutional layer.

**Definition 3.1.14** (convolutional layer). The pair  $(\Phi_{conv}^{\mathbf{K}, \mathbf{b}}, \psi)$  of convolution logit function and activation function is called *convolutional layer*. For a given *input*  $\mathbf{X} \in \mathbb{R}^{* \times * \times z_{in}}$  the layer's *output*  $\mathbf{Y} \in \mathbb{R}^{* \times * \times z_{out}}$  is calculated as follows

$$\mathbf{Y} = \psi \circ \Phi_{conv}^{\mathbf{K}, \mathbf{b}}(\mathbf{x}).$$

The matrices  $\mathbf{Y}_{::, k}$ ,  $k \in [z_{out}]$  are called *feature maps*.

For the reason of simplicity, we usually write  $\Phi_{conv}^{\mathbf{K}, \mathbf{b}, \psi}$  for a layer and  $\Phi_{conv}^{\mathbf{K}, \mathbf{b}, \psi}(\mathbf{X}) = \psi(\Phi_{conv}^{\mathbf{K}, \mathbf{b}}(\mathbf{X}))$  holds. The convolutional layer as introduced in this thesis retains the spatial dimensions of the input. This is due to the fact that in Def. 3.1.10 zero padding is performed at the border. In the literature, convolutional layers sometimes do not perform zero padding. Hence, the spatial dimensions of the input are reduced



by the kernel size minus one (in each dimension), if we consider a stride of 1 along each dimension.

Besides these border effects, one usually aims at the reduction of the spatial dimensions and an increase of the representative depth to build complex, high-level and more global features as input for subsequent convolutional layers. This reduction of the spatial dimensions is usually done by so-called pooling layers. Wen et al. [WAH92] introduce the max pooling layer. Ranzato et al. [Ran+07] use the max pooling layer in conjunction with convolutional layers. Nowadays, the max pooling layer is the widely used standard in many state-of-the-art systems and is also used within this work.

**Definition 3.1.15** (max pooling layer). For  $p_h, p_w \in \mathbb{N}$  the function  $\Phi_{max}^{p_h, p_w} : \mathbb{R}^{* \times * \times z} \rightarrow \mathbb{R}^{* \times * \times z}$  defined as

$$\Phi_{max}^{p_h, p_w}(\mathbf{X})_{i,j,k} := \max_{\substack{(i-1) \cdot p_h < i' \leq \min\{i \cdot p_h, 1 \cdot \mathbf{X}\} \\ (j-1) \cdot p_w < j' \leq \min\{j \cdot p_w, 2 \cdot \mathbf{X}\}}} \mathbf{X}_{i',j',k} \quad \forall i \in [\lceil 1 \cdot \mathbf{X} / p_h \rceil], \\ \forall j \in [\lceil 2 \cdot \mathbf{X} / p_w \rceil], \quad \forall k \in [z]$$

is called *max pooling layer*.  $p_h, p_w$  are called *subsampling factors*.

Basically, the max pooling layer subsamples an input by reducing the information of all values within a *receptive field* (which is the spatial region of interest for a certain computation) of dimension  $p_h \times p_w$  to their maximum value. The depth is unchanged.

To allow for the combination of classical layers and convolutional layers a flatten function is introduced.

**Definition 3.1.16** (flatten function). For any  $n_1, n_2, n_3$  let  $f_f : \mathbb{R}^{n_1 \times n_2 \times n_3} \rightarrow \mathbb{R}^{n_1 \cdot n_2 \cdot n_3}$  be the function mapping any  $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$  to an  $(n_1 \cdot n_2 \cdot n_3)$ -dim vector in the following way

$$f_f(\mathbf{X})_{(i-1) \cdot (n_2 \cdot n_3) + (j-1) \cdot n_3 + k} := \mathbf{X}_{i,j,k} \quad \forall i \in [n_1], j \in [n_2], k \in [n_3]$$

This function is called *flatten function*.

Based on the introduced termini and concepts the definition of a CNN can be formulated as follows.

**Definition 3.1.17** (convolutional neural network). For  $h, w, z_{in}, z_{out}, l, c \in \mathbb{N}$  we call a sequence of convolutional layers  $\Phi_{conv}^{K^{(1)}, \mathbf{b}^{(1)}, \psi_1}, \dots, \Phi_{conv}^{K^{(c)}, \mathbf{b}^{(c)}, \psi_c}$  and a sequence of classical layers  $\Phi^{W^{(1)}, \mathbf{b}^{(1)}, \psi_1}, \dots, \Phi^{W^{(l)}, \mathbf{b}^{(l)}, \psi_l}$  *convolutional neural network (CNN)* iff for  $\mathbf{X} \in \mathbb{R}^{h \times w \times z_{in}}$  the *forward propagation*

$$\mathbf{y} = \Phi^{W^{(l)}, \mathbf{b}^{(l)}, \psi_l} \circ \dots \circ \Phi^{W^{(1)}, \mathbf{b}^{(1)}, \psi_1} \circ f_f \circ \Phi_{conv}^{K^{(c)}, \mathbf{b}^{(c)}, \psi_c} \circ \dots \circ \Phi_{conv}^{K^{(1)}, \mathbf{b}^{(1)}, \psi_1}(\mathbf{X}) \quad (3.1.4)$$

is well-defined. That means, the MLP condition (Eq. (3.1.1)) holds for the set of classical layers with an input dimension equal to the output dimension of  $f_f$  and an output dimension of  $z_{out}$ . In Eq. (3.1.4) each convolutional layer can be composed with a max pooling layer.

The I/O dimensions  $h, w, z_{in}, z_{out}$ , the number of convolutional layers  $c$  and classical layers  $l$ , the dimensionality of the kernels, weights, and biases, the number/position/size of max pooling layers as well as the choice of the activation functions are called *hyper parameters* of the CNN. The kernels, weights, and biases themselves constitute the *model parameters*.

**Remark 3.1.18.** A CNN is a function  $\Phi(\cdot; \theta) : \mathbb{R}^{h \times w \times z_{in}} \rightarrow \mathbb{R}^{z_{out}}$ ,  $\theta$  constitutes the model parameters.

Traditionally, the CNN is used for "global" classification purposes. In this context global means that for an input a single (global) output for the entire input is generated, e.g., digit classification [CMS12], optical character recognition [LeC+98], image classification [KSH12]. Nevertheless, the CNN still suffers from a fixed input dimension due to the transition between convolutional layers and classical layers (cf. Eq. (3.1.4)).

This is solved by the *fully convolutional network (FCN)*. A FCN is a CNN without flatten function and classical layers. I.e., the FCNs contain only convolutional and max pooling layers. Hence, the MLP condition (Eq. (3.1.1)) does not have to hold and the spatial input dimension is arbitrary. Furthermore, the output of the FCN is still a 3-dim array.

**Remark 3.1.19.** An FCN is a function  $\Phi(\cdot; \theta) : \mathbb{R}^{* \times * \times z_{in}} \rightarrow \mathbb{R}^{* \times * \times z_{out}}$ ,  $\theta$  constitutes the model parameters.

The FCNs are recently very popular for pixel labeling/semantic segmentation tasks [LSD15]; [NHH15]; [RFB15]; [Che+17]; [Ten+17]; [Vo+17]. In Sec. 4.3.1 we will introduce the ARU-Net which is the basis for the proposed baseline detection method. This ARU-Net is basically a FCN.

### 3.1.3. Tasks & Loss Functions for ANNs

In the last sections, we have introduced the MLP, the CNN, and the FCN. All of them are representatives of ANNs which themselves belong to the class of machine learning algorithms. But what exactly is meant by learning? One answer to this question was given in [Mit97]: *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.* In this section we will

focus on tasks (T) and measures (P) for ANNs. The experience (E) is discussed in Sec. 3.1.4.

There are various different tasks, e.g., classification, regression, transcription, machine translation, anomaly detection, denoising, which were tackled with ANN based systems in the last years. In Sec. 3.1, several examples were given for systems designed to solve some of the above stated tasks. Of course, there are at least as many measures as there are tasks. In this thesis, we will limit ourselves to the classification task and a suitable measure. The classification task was extensively studied in machine learning literature [MBi06]; [DHS12]. In this work, we will briefly introduce the classification task and a suitable loss function for the MLP scenario and extend it to the CNN & FCN scenario in the end.

### Classification Task

**Definition 3.1.20** (classification function). Let  $\mathbb{X} \subset \mathbb{R}^n$  and  $\mathfrak{C} = \{c_1, \dots, c_m\}$  be a set of distinct *classes* ( $c_i$  is called *class label*). A function  $f_C : \mathbb{X} \rightarrow \mathfrak{C}$  which assigns a class label to each element of  $\mathbb{X}$  is called *classification function*.

The classification task aims at the approximation of a certain classification function  $f_C$  by an appropriate model. There are mainly three different approaches to do so, see [MBi06] [pp. 43]. The first is to build a model  $\tilde{f}_C : \mathbb{X} \rightarrow \mathfrak{C}$  that directly maps from the input space to a certain output class label, this type of model is referred to as *discriminant function*. The support vector machine [CV95] is a well-known example of this kind.

The other two classes of models are *probabilistic models* and incorporate some uncertainty in terms of a joint (probability) distribution  $p_{data}(\mathbf{x}, c)$  defined on the sample space  $\Omega = \mathbb{X} \times \mathfrak{C}$ . We refer to this distribution as *data distribution*. Hence, the entire uncertainty associated to the input and class variables is encoded by  $p_{data}(\mathbf{x}, c)$ . To solve the classification task, the models are designed to estimate the *posterior distribution*  $p_{data}(\cdot | \mathbf{x})$  of the classes given an input.

The *generative models* try to separately infer the class-conditional probabilities  $p_{data}(\mathbf{x} | c)$  and the prior class probabilities  $p_{data}(c)$  and utilize Bayes' theorem to estimate the posterior probabilities for all classes. The hidden markov models [Rab89] are a representative of this type of models. However, the ANNs as introduced in this thesis are *discriminative models* and we will limit to this kind of models in this work. This type of models tries to infer the posterior distribution  $p_{data}(\cdot | \mathbf{x})$  given an input  $\mathbf{x}$  directly. Hence, one aims for the approximation of the function  $P_{data} : \mathbb{X} \rightarrow [0, 1]^m$  defined as

$$P_{data}(\mathbf{x}) := (p_{data}(c_1 | \mathbf{x}), \dots, p_{data}(c_m | \mathbf{x}))^T \quad \forall \mathbf{x} \in \mathbb{X}. \quad (3.1.5)$$

Let

$$P_{model}(\mathbf{x}; \boldsymbol{\theta}) := (p_{model}(c_1 | \mathbf{x}; \boldsymbol{\theta}), \dots, p_{model}(c_m | \mathbf{x}; \boldsymbol{\theta}))^T \quad \forall \mathbf{x} \in \mathbb{X} \quad (3.1.6)$$

be the model's approximation of  $P_{data}$  which is parametrized by  $\boldsymbol{\theta} \in \mathbb{R}^N$ . The classification result can be calculated as

$$f_{model}(\mathbf{x}) := \arg \max_{c \in \mathcal{C}} p_{model}(c | \mathbf{x}). \quad (3.1.7)$$

This separation of inference and decision (Eq. (3.1.6) & (3.1.7)) has several advantages over the approach of discriminant functions, see [MBi06] Sec. 1.5.4. Note that  $p_{model}$  constitutes a family of distributions over the classes which is dependent on the model input.

In the following, we will examine whether the MLP is a kind of model which is capable to approximate any  $P_{data}$  (and is therefore a suitable choice) or not. First, one can show that the MLP is a universal function approximator.

**Theorem 3.1.21** (universal approximation theorem). *Let  $\psi_1 : \mathbb{R} \rightarrow \mathbb{R}$  be a non-constant, bounded, and monotonically-increasing continuous activation function.  $id : \mathbb{R} \rightarrow \mathbb{R}$  is the identity and  $\mathbb{X} \subset \mathbb{R}^n$  is compact. For any  $\epsilon > 0$  and any continuous function  $f : \mathbb{X} \rightarrow \mathbb{R}$ , there exist  $N \in \mathbb{N}$ ,  $W^{(1)} \in \mathbb{R}^{n \times N}$ ,  $\mathbf{b}^{(1)} \in \mathbb{R}^N$ , and  $W^{(2)} \in \mathbb{R}^{N \times 1}$  such that*

$$\left| f(\mathbf{x}) - \Phi^{W^{(2)}, \mathbf{0}, id} \circ \Phi^{W^{(1)}, \mathbf{b}^{(1)}, \psi_1}(\mathbf{x}) \right| < \epsilon \quad \forall \mathbf{x} \in \mathbb{X} \quad (3.1.8)$$

holds.

*Proof.* see [Cyb89]; [Hor91]; [Has95] □

The restriction to non-constant and bounded activation functions is not contemporary because the relu activation, which is constant (for  $x < 0$ ) and not bounded, is used in many state-of-the-art systems. Sonoda et al. [SM17] extend the universal approximation theorem for more general activation functions, especially for the relu activation function.

**Corollary 3.1.22.** *The universal approximation theorem holds for the relu activation  $\psi_1(x) = \max\{0, x\}$ .*

*Proof.* see [SM17] □

The universal approximation theorem can easily be extended to functions  $f : \mathbb{X} \rightarrow \mathbb{R}^m$ . But in this generic extension the activation functions are decoupled, i.e., the activations are calculated for each dimension independently. Hence, it does not hold for the softmax activation. This was investigated by Strauß [Str17].

**Corollary 3.1.23.** *A 2-layer MLP with a softmax activation for the output layer can approximate (to arbitrary precision) any continuous function ( $P_{data}$ ) mapping from a compact domain to a (probability) distribution over the possible output classes.*

*Proof.* see [Str17] □

It is easy to see that this collary is still true for any  $n$ -layer MLP. For practically relevant problems  $\mathbb{X}$  of Def. 3.1.20 is usually finite, e.g., the finite set of (relevant) images in 8-bit resolution, the finite set of digitized audio signals. As a result,  $\mathbb{X}$  is compact and  $P_{data}$  is continuous. Therefore, the MLP is capable to approximate  $P_{data}$  to arbitrary precision and is an appropriate discriminative model.

After showing that the MLP is a suitable discriminative model, we focus on the measure of the quality of the approximation of  $P_{data}$  by  $P_{model}$ . The quality will be measured in terms of a *loss function*. Because  $\Omega = \mathbb{X} \times \mathfrak{C}$  is usually intractably large, let

$$\mathfrak{T}' = \{(\mathbf{x}^{(i)}, c^{(i)}) \mid i \in [n_{tr}]\} \subset \mathbb{X} \times \mathfrak{C} \quad (3.1.9)$$

be a training set which is i.i.d. sampled from the data distribution  $p_{data}$ . The quality of  $P_{model}$  can now be evaluated in terms of how probable it is to independently sample the correct classes of the training set from the model. The resulting probability is called the likelihood of the model (given the training data) and is given by

$$L(\mathfrak{T}'; \boldsymbol{\theta}) = \prod_{(\mathbf{x}, c) \in \mathfrak{T}'} p_{model}(c \mid \mathbf{x}; \boldsymbol{\theta}) \in [0, 1],$$

see [MBi06]; [Gra08]; [GBC16]. Traditionally, one aims for a large likelihood, i.e., the product over the probabilities for the correct class labels for all samples in  $\mathfrak{T}'$  should be large. Consequently, the maximum likelihood approach tries to solve

$$\boldsymbol{\theta}^* = \arg \max_{\boldsymbol{\theta} \in \mathbb{R}^N} \prod_{(\mathbf{x}, c) \in \mathfrak{T}'} p_{model}(c \mid \mathbf{x}; \boldsymbol{\theta}).$$

Obviously, this is equivalent to a maximization of the log likelihood

$$\boldsymbol{\theta}^* = \arg \max_{\boldsymbol{\theta} \in \mathbb{R}^N} \log \prod_{(\mathbf{x}, c) \in \mathfrak{T}'} p_{model}(c \mid \mathbf{x}; \boldsymbol{\theta}) = \arg \max_{\boldsymbol{\theta} \in \mathbb{R}^N} \sum_{(\mathbf{x}, c) \in \mathfrak{T}'} \log p_{model}(c \mid \mathbf{x}; \boldsymbol{\theta}). \quad (3.1.10)$$

The latter maximization problem does not suffer from numerical underflow errors due to the product of many probabilities.

Let  $(\mathbf{x}, c) \in \mathfrak{T}'$  be a training sample. We encode the class  $c$  in an *one-hot encoding* scheme (or 1-of- $m$  coding scheme) by  $\mathbf{c} \in \mathbb{R}^m$  ( $m$  is the number of possible classes)

which is defined component-wise as

$$\mathbf{c}_i := \begin{cases} 1 & , \text{ if } c = c_i \\ 0 & , \text{ else} \end{cases} \quad \forall i \in [m].$$

As a result, we can formulate the negative log likelihood loss function in the notation used within this thesis. It is easy to see that its minimization is equivalent to the maximization of Eq. (3.1.10).

**Definition 3.1.24** (negative log likelihood). Let  $\mathbb{X}, \mathfrak{C} = \{c_1, \dots, c_m\}$  be as in Def. 3.1.20 and  $\mathfrak{T}'$  as in Eq. (3.1.9). Let  $p_{model}$  denote the model posterior probabilities defined in Eq. (3.1.6). The function  $L_{NLL}$  defined as

$$L_{NLL}(p_{model}; \mathfrak{T}') := - \sum_{(\mathbf{x}, c) \in \mathfrak{T}'} \sum_{i=1}^m \mathbf{c}_i \cdot \log p_{model}(c_i \mid \mathbf{x}; \boldsymbol{\theta}).$$

is called *negative log likelihood* ( $L_{NLL}$ ).

Besides the motivation for  $L_{NLL}$  which is based on the maximization of the likelihood of the model given the training data, we will give another motivation for this loss function. The second motivation is based on the well-known (discrete) Kullback-Leibler divergence [KL51] which is a measure for the disagreement between two distributions.

For each  $(\mathbf{x}, c) \in \mathfrak{T}'$  let  $p_{gt}(\cdot; \mathbf{x}, c) : \mathfrak{C} \rightarrow \{0, 1\}$  be the GT distribution which is defined as

$$p_{gt}(\tilde{c}; \mathbf{x}, c) := \begin{cases} 1 & , \text{ if } \tilde{c} = c \\ 0 & , \text{ else} \end{cases}.$$

Consequently, we can measure the disagreement between the GT distribution and the model distribution for a certain trainings sample  $(\mathbf{x}, c)$  utilizing the (discrete) Kullback-Leibler divergence as

$$\begin{aligned} D_{KL}(p_{gt} \parallel p_{model}; \mathbf{x}, c) &= \sum_{i=1}^m p_{gt}(c_i; \mathbf{x}, c) \cdot \log \frac{p_{gt}(c_i; \mathbf{x}, c)}{p_{model}(c_i \mid \mathbf{x}; \boldsymbol{\theta})} \\ &= \sum_{i=1}^m p_{gt}(c_i; \mathbf{x}, c) \cdot (\log p_{gt}(c_i; \mathbf{x}, c) - \log p_{model}(c_i \mid \mathbf{x}; \boldsymbol{\theta})) \\ &= - \sum_{i=1}^m p_{gt}(c_i; \mathbf{x}, c) \cdot \log p_{model}(c_i \mid \mathbf{x}; \boldsymbol{\theta}) \\ &= - \sum_{i=1}^m \mathbf{c}_i \cdot \log p_{model}(c_i \mid \mathbf{x}; \boldsymbol{\theta}). \end{aligned}$$

The last but one equality holds, because of the one-hot encoding of  $p_{gt}$  and because the summands  $0 \cdot \log 0$  count 0 in the sum (motivated by  $\lim_{x \rightarrow 0} x \cdot \log x = 0$ ). Of

course, the disagreement should not be penalized for a single training sample but for the entire training set. Finally, the summation over the elements in  $\mathfrak{T}'$

$$D_{KL}(p_{gt} \parallel p_{model}; \mathfrak{T}') = - \sum_{(x,c) \in \mathfrak{T}'} \sum_{i=1}^m \mathbf{c}_i \cdot \log p_{model}(c_i \mid \mathbf{x}; \boldsymbol{\theta}).$$

yields the  $L_{NLL}$ .

**Remark 3.1.25.** The  $L_{NLL}$  as introduced in this section can be used to approximate any  $p_{data}$ . Nonetheless, the data distribution resulting from the classification function of Def. 3.1.20 is one-hot encoded. This is due to the fact that  $f_C$  is a function.

**Remark 3.1.26.** Originally, one aims at the approximation of  $p_{data}$  for the entire sample space. In contrast, the loss function is minimized for a certain subset of the sample space ( $\mathfrak{T}'$ ). Hence, usually another subset  $\mathfrak{T}$  (the test set) is sampled from the sample space to get a more reliable estimation of the quality of the model, see Sec. 3.1.4.

### Spatial Classification Task

The introduced concepts are now extended to the spatial case. This case allows for a pixel labeling given an input image as it will be shown in a while. In the following, the I/O arrays are 3-dimensional in the spatial case (two spatial dimensions and one for the representative depth). First, the  $r$ -interior is introduced to handle border effects.

**Definition 3.1.27** ( $r$ -interior). Let  $\mathbf{X} \in \mathbb{R}^{h \times w \times n}$  and  $r \in \mathbb{N}$ . The 3-dim array  $\mathbf{Y} \in \mathbb{R}^{h-2 \cdot r \times w-2 \cdot r \times n}$  is called  $r$ -interior of  $\mathbf{X}$  iff

$$\mathbf{Y}_{i,j,k} = \mathbf{X}_{i+r,j+r,k} \quad \forall i \in [h-2 \cdot r], j \in [w-2 \cdot r], k \in [n].$$

**Definition 3.1.28** (spatial classification function). Let  $\mathbb{X} \subset \mathbb{R}^{* \times * \times n}$  and  $\mathfrak{C} = \{c_1, \dots, c_m\}$  be a set of distinct classes. We call a function  $f_{SC} : \mathbb{X} \rightarrow \mathfrak{C}^{* \times *}$  *spatial classification function* iff

- (1)  ${}_1 f_{SC}(\mathbf{X}) = {}_1 \mathbf{X} \wedge {}_2 f_{SC}(\mathbf{X}) = {}_2 \mathbf{X} \quad \forall \mathbf{X} \in \mathbb{X}$
- (2)  $f_{SC}$  has a receptive field of radius  $r \in \mathbb{N}$ , i.e.,  $f_{SC}(\mathbf{X})_{i,j} = f_{SC}(\widetilde{\mathbf{X}})_{i,j}$  holds if  $\mathbf{X}, \widetilde{\mathbf{X}}$  are equal in a circle of radius  $r$  around  $(i, j)$

hold.

The first condition states that the spatial dimensions of input and output are equal. That means a spatial classification function assigns a class to each spatial point of the input. The second condition implies that the classification result at a certain

position is just influenced by the input information in a field of finite spatial radius around the position of interest.

Now, the spatial classification task is about the approximation of  $f_{SC}$  by a certain model. In the literature, the spatial classification task is often referred to as image segmentation [PP93], semantic segmentation [LSD15], or pixel labeling [NW12]. For the spatial classification task, we also follow the approach of discriminative models. The introduced distribution of Eq. (3.1.5) is extended to the spatial case in the generic way and is denoted by  $P_{data}^S$ . First, we will prove the capability of the FCN model to solve the spatial classification task.

**Corollary 3.1.29.** *Let  $f_{SC}$  be a spatial classification function defined on a finite domain  $\mathbb{X} \subset \mathbb{R}^{* \times * \times n}$  and  $r \in \mathbb{N}$  is the radius of its receptive field.  $P_{data}^S$  denotes the spatial posterior distribution over the distinct classes  $\mathfrak{C} = \{c_1, \dots, c_m\}$ . There is a 2-layer FCN with a convolutional softmax activated output layer that approximates  $P_{data}^S$  in the  $r$ -interior to arbitrary precision.*

*Proof.* Let  $\tilde{\mathbb{X}} \subset \mathbb{R}^{(2 \cdot r + 1) \times (2 \cdot r + 1) \times n}$  be the set of all  $(2 \cdot r + 1) \times (2 \cdot r + 1)$  patches  $((2 \cdot r + 1) \times (2 \cdot r + 1) \times n$  dimensional subarrays) of elements of  $\mathbb{X}$ . Because  $f_{SC}$  is equivariant with respect to translations on the  $r$ -interior, it is sufficient to prove the claim (w.l.o.g.) for the mapping  $\widetilde{f_{SC}} : \tilde{\mathbb{X}} \rightarrow \mathfrak{C}$  which is defined as

$$\widetilde{f_{SC}}(\tilde{\mathbf{X}}) := f_{SC}(\mathbf{X})_{i,j} \quad (3.1.11)$$

for some  $\mathbf{X} \in \mathbb{X}$  for which  $\tilde{\mathbf{X}}$  is patch of  $\mathbf{X}$  centered at  $i, j$ . Let  $f_f^{-1}$  be the inversion of the flatten function (cf. Def. 3.1.16) for the fixed spatial dimensions of  $\tilde{\mathbb{X}}$ . There is an MLP approximating the posterior distribution for the classification problem resulting from  $f_C : \mathbb{R}^{(2 \cdot r + 1)^2 \cdot n} \rightarrow \mathfrak{C}$

$$f_C(\mathbf{x}) := \widetilde{f_{SC}}(f_f^{-1}(\mathbf{x})) \quad (3.1.12)$$

to arbitrary precision, cf. Cor. 3.1.23. Let  $W^{(1)} \in \mathbb{R}^{(2 \cdot r + 1)^2 \cdot n \times N}$ ,  $\mathbf{b}^{(1)} \in \mathbb{R}^N$ ,  $W^{(2)} \in \mathbb{R}^{N \times m}$  be the model parameters of this MLP.  $\psi_1$  is the activation function of the first layer. Let  $\mathbf{K}^{(1)} \in \mathbb{R}^{2 \cdot r + 1 \times 2 \cdot r + 1 \times n \times N}$  be defined by  $W^{(1)}$  via a reordering of its elements along the first dimension determined by  $f_f^{-1}$ . Furthermore, let  $\mathbf{K}^{(2)} \in \mathbb{R}^{1 \times 1 \times N \times m}$  be defined by  $\mathbf{K}_{1,1,,:}^{(2)} = W^{(2)}$ . Let  $\psi_2$  be the spatial softmax activation function. The 2-layer FCN defined as

$$\mathbf{Y} = \Phi_{conv}^{\mathbf{K}^{(2)}, \mathbf{0}, \psi_2} \circ \Phi_{conv}^{\mathbf{K}^{(1)}, \mathbf{b}^{(1)}, \psi_1}(\mathbf{X}) \quad (3.1.13)$$

approximates  $P_{data}^S$  in the  $r$ -interior to arbitrary precision.  $\square$

Because the arguments given for the classification task on practically relevant problems still hold for the spatial classification task, the FCN has the capability to solve



the spatial classification task. Finally, the negative log likelihood function is extended to the spatial case. Of course, its motivations are also valid for the spatial case.

**Definition 3.1.30** (spatial negative log likelihood). Let  $\mathbb{X}, \mathfrak{C}$  be as in Def. 3.1.28,  $\mathfrak{T}' = \{(\mathbf{X}^{(i)}, C^{(i)}) \mid i \in [n]\} \subset \mathbb{X} \times \mathfrak{C}^{* \times *}$  is a training set as in Eq. (3.1.9), and  $\mathbf{C}$  is the one-hot encoded version of  $C$ .  $p_{model}^S$  denotes the spatial posterior distribution of the model for a certain input, cf. Eq. (3.1.6). The function  $L_{SNLL}$  defined as

$$L_{SNLL}(p_{model}^S; \mathfrak{T}') = - \sum_{(\mathbf{X}, C) \in \mathfrak{T}'} \sum_{i=1}^{1\mathbf{X}} \sum_{j=1}^{2\mathbf{X}} \sum_{k=1}^m \mathbf{C}_{i,j,k} \cdot \log p_{model}^S(c_k \mid \mathbf{X})_{i,j}. \quad (3.1.14)$$

is called *spatial negative log-likelihood* ( $L_{SNLL}$ ).

$L_{SNLL}$  will be used as loss function throughout this thesis.

### 3.1.4. Training of ANNs

Roughly speaking, this section is about the realization of the theoretical capability to solve the classification (and other) tasks. The corollaries of the last section just stated that "there is a MLP/FCN that ...", they did not give any idea how such models look like. In this section, we focus on the estimation of the model parameters, such that a certain model approximates the underlying data distribution well. In this work, this problem is tackled following the paradigm of *supervised learning*, see [MBi06] [pp. 3], [GBC16] [pp. 104]. As a result, the experience (E) of the last section constitutes the learning of the model parameters given a labeled training set.

In this section, we will restrict the discriminative models approximating the posterior probabilities over the output classes  $\Phi(\cdot; \boldsymbol{\theta}) : \mathbb{X} \rightarrow [0, 1]^m$  with  $\mathbb{X} \subset \mathbb{R}^n$  to MLPs. The extension to FCNs is straightforward. The MLP maps any input vector to a distribution over all output classes. The model is parametrized by  $\boldsymbol{\theta}$ . Hence,  $\boldsymbol{\theta} \in \mathbb{R}^N$  contains all model parameters (weights and biases). Assume there are two distinct sets

$$\begin{aligned} \mathfrak{T} &= \{(\mathbf{x}^{(i)}, \mathbf{c}^{(i)}) \mid i \in [n_{test}], \mathbf{x}^{(i)} \in \mathbb{X}, \mathbf{c}^{(i)} = P_{data}(\mathbf{x}^{(i)})\} - \text{test set}, \\ \mathfrak{T}' &= \{(\mathbf{x}^{(i)}, \mathbf{c}^{(i)}) \mid i \in [n_{tr}], \mathbf{x}^{(i)} \in \mathbb{X}, \mathbf{c}^{(i)} = P_{data}(\mathbf{x}^{(i)})\} - \text{train set}, \end{aligned}$$

of input vectors along with their classification results ( $P_{data}$  of Eq. (3.1.5)). The examples are i.i.d. sampled from  $\mathbb{X}$ . In supervised learning, the model parameters should be estimated to minimize the loss function  $L$ , e.g., the negative log-likelihood of Def. 3.1.24, on the training set.

**Definition 3.1.31** (error function). The function  $\mathcal{E}(\cdot; \mathfrak{T}') : \mathbb{R}^N \rightarrow \mathbb{R}$  defined as

$$\mathcal{E}(\boldsymbol{\theta}; \mathfrak{T}') := \frac{1}{|\mathfrak{T}'|} \sum_{(\mathbf{x}, \mathbf{c}) \in \mathfrak{T}'} L(\Phi(\mathbf{x}; \boldsymbol{\theta}), \mathbf{c})$$

is called *error function*.

Hence, in supervised learning the minimization problem

$$\mathcal{E}(\boldsymbol{\theta}; \mathfrak{T}') \rightarrow \min \tag{3.1.15}$$

in the parameters  $\boldsymbol{\theta}$  has to be solved. The process of estimating the optimal model parameter is referred to as *training*.

After solving optimization problem (3.1.15), the quality of the model is evaluated on the test set. Ideally, the training process results in the generalization to new, previously unseen data (test set). It is not about learning solely the training data. The differences and implications of this are described in detail in [GBC16] [Sec. 5.2]. To overcome the problem of memorizing solely the training data (this is referred to as *overfitting*) instead of approximating the underlying data distribution, one usually uses large training datasets [GBC16] [pp. 152] or employs data augmentation strategies to increase the number of training samples, see Sec. 4.4.

Because of the complexity of  $\mathcal{E}$  and the large number of parameters ( $N$  usually exceeds  $1M$ ), there is no direct solution of optimization problem (3.1.15). Indeed, Blum et al. [BR89] show that even the training of a 3-node neural network is NP-complete. Instead, methods based on *gradient descent* [Cau47]; [Had08] are usually utilized to iteratively and approximately solve optimization problem (3.1.15). There are also other techniques to estimate the model parameters of an ANN [Heb49]; [Jae02], but they are usually limited to special scenarios and not competitive in general.

To apply gradient descent, the function  $\mathcal{E}$  has to be at least piecewise continuously differentiable. The MLP (FCN) along with the  $L_{NLL}$  ( $L_{SNLL}$ ) as introduced in this the last section are piecewise continuously differentiable since the introduced components (see Sec. 3.1.1-3.1.3) are all piecewise continuously differentiable. Consequently, their composition (MLP/FCN) is also piecewise continuously differentiable. The calculation of the error function's gradient with respect to the model parameters  $\nabla_{\boldsymbol{\theta}} \mathcal{E}(\boldsymbol{\theta}; \mathfrak{T}')$ , especially for deep structures, is challenging. The technique of (*error*) *backpropagation* or *backward propagation of errors* iterates the chain rule [Lei76]; [LH696] to efficiently calculate this gradient.

Its invention was a milestone in deep learning. Traditionally, the work of Rumelhart et al. [RHW86] is cited in conjunction with backpropagation. Nevertheless, the minimisation of errors through gradient descent for ANN-related systems has been already discussed in the 1960s [Bry61]; [BH69]. Efficient error backpropaga-

tion in arbitrary, discrete, possibly sparsely connected, ANN-like networks was first described in the 1970s [Lin70]; [Lin76]. Afterwards, in the 1980s efficient backpropagation was proposed for ANN-specific applications [Wer81]; [Par85]; [LeC85]. Finally, Rumelhart et al. [RHW86] significantly contributed to the popularisation of backpropagation for ANNs. They experimentally demonstrated the benefit of internal representations in hidden layers which were trained by backpropagation. Error backpropagation was formulated for CNNs (and consequently for FCNs) by LeCun et al. [LeC+89]; [LeC+90]; [LeC+98]. Hence, we can assume that  $\nabla_{\theta}\mathcal{E}(\theta; \mathfrak{T}')$  is calculable and gradient descent is applicable.

Let  $\theta^{(i)}$  denote the model parameters after optimization step  $i$ . The update rule for the *batch gradient descent* can now be formulated as

$$\theta^{(i+1)} = \theta^{(i)} - \eta \cdot \nabla_{\theta}\mathcal{E}(\theta^{(i)}; \mathfrak{T}')$$

with a *learning rate*  $\eta \in \mathbb{R}_+$ . Since the gradient comprises the entire training set, which is typically large, a single gradient descent step is computationally expensive. As a consequence, the widely used standard to solve the optimization problem stated in (3.1.15) is the *stochastic gradient descent*. In stochastic gradient descent the gradient  $\nabla_{\theta}\mathcal{E}(\theta; \mathfrak{T}')$  is approximated by the gradient for a subset  $\mathbb{T} \subset \mathfrak{T}'$  containing just a few, e.g.,  $|\mathbb{T}| = 16, 32, 64$ , training samples. We call a subset  $\mathbb{T} \subset \mathfrak{T}'$  *minibatch*. The minibatch can be sampled with or without returning of the chosen samples. In this work, we sample without returning. Finally, we refer to the optimization as *minibatch gradient descent (MGD)* and the training follows Alg. 1.

---

**Algorithm 1:** Minibatch Gradient Descent:  $\theta = \text{MGD}(\mathcal{E}, \mathfrak{T}', \theta_0, \eta, n)$

---

**input** : error function  $\mathcal{E}$ , training set  $\mathfrak{T}'$ , initial value  $\theta_0$ , learning rate  $\eta$ , minibatch size  $n$   
**output**: optimized model parameter  $\theta$

```

1  $\theta \leftarrow \theta_0, \mathbb{T} \leftarrow \mathfrak{T}'$ 
2 while  $\neg \text{STOP}$  do ▷ stopping criteria, see main text
3   if  $|\mathbb{T}| < n$  then
4      $\mathbb{T} \leftarrow \mathfrak{T}'$ 
5    $\mathbb{T}_n \leftarrow$  randomly choose  $n$  samples of  $\mathbb{T}$ 
6    $\mathbb{T} \leftarrow \mathbb{T} \setminus \mathbb{T}_n$ 
7    $\theta \leftarrow \theta - \eta \cdot \nabla_{\theta}\mathcal{E}(\theta; \mathbb{T}_n)$ 
return :  $\theta$ 
```

---

There are various stopping criteria. The simplest is to train for a fixed pre-defined number of update steps. Hence, the stopping criteria is just a counter. The second approach is to implement some early stopping [MBi06] [pp. 259] based on, e.g., the gradient's norm, the training error or the error on a so-called validation set, see [GBC16] [pp. 246].

In practice, this vanilla version of MGD often yields bad results. Especially, for deep architectures with several stacked layers the results are not satisfying at all. The training suffers from vanishing gradient phenomena [GB10] and/or is prone to local minima as well as saddle points [Dau+14]. This is the main reason, why researchers lost interest in ANNs in the 1990s and early 2000s. A breakthrough in *unsupervised learning* [HOT06]; [HS06]; [Ben+07] was necessary to unleash the potential of the ANNs. Deep fully connected ANNs which were pre-trained in a fully unsupervised fashion outperformed the previous state of the art, e.g., SVMs, on some benchmarking problems like MNIST [LeC98]. This was the beginning of the renaissance of ANNs. Since that time, substantial progress happened concerning the training of deep ANNs. Nowadays, deep neural networks (and even deep recurrent neural networks) are trainable without any pre-training yielding amazing results.

Besides the classical method of momentum [Pol64], there are various first order optimization methods which were proposed in the last years, e.g., Nesterov accelerated gradient [Sut+13], AdaGrad [DHS11], RMSProp [TH12], AdaDelta [Zei12], Adam [KB15], Nadam [Doz16], to name but a few. They basically adjust line 7 in Alg. 1. We will shortly introduce *RMSPProp* because this is the training scheme used within this work.

**Definition 3.1.32** (exponential moving average). Let  $\mathbf{x}^{(t)} \in \mathbb{R}^N$  be a time dependent  $t = 1, 2, \dots$  series of vectors. The recursively defined value

$$EMA(\mathbf{x})^{(t)} = (1 - \xi) \cdot \mathbf{x}^{(t)} + \xi \cdot EMA(\mathbf{x})^{(t-1)} \quad t = 2, 3, \dots$$

with an initial value of  $EMA(\mathbf{x})^{(1)} = \mathbf{x}^{(1)}$  and  $\xi \in [0, 1]$  is called *exponential moving average* (EMA).

The EMA results in a smoothing of high frequencies (for  $\xi$  close to one) of the underlying time series of vectors. This is utilized to get a (squared) gradient estimation in MGD which is not prone to the noise because of sampling issues. This estimate is used to rescale the gradient component-wise to tackle the vanishing/exploding gradient phenomena. Finally, for RMSProp line 7 in Alg. 1 becomes:

---


$$\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta}} \mathcal{E}(\boldsymbol{\theta}; \mathbb{T}_n)$$

$$EMA(\mathbf{g}^2) \leftarrow (1 - \xi) \cdot \mathbf{g}^2 + \xi \cdot EMA(\mathbf{g}^2) \quad \triangleright \mathbf{g}^2 \text{ element-wise square}$$

$$\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i - \frac{\eta}{\sqrt{EMA(\mathbf{g}^2)_i + 10^{-8}}} \cdot \mathbf{g}_i, \quad i \in [|\boldsymbol{\theta}|]$$


---

Consequently, each weight gets an own learning rate which depends on the norm of its gradient's exponential moving average. A parametrization of  $\xi = 0.9$  is recommended for the newly introduced hyperparameter.

Furthermore, several different techniques for weight initialization ( $\boldsymbol{\theta}_0$ ) were proposed [GB10]; [SMG13]; [Sus15]; [Seu+17]. In this work, we utilize the weight initialization

of [GB10]. It is often referred to as Xavier initialization. Basically, the weights are sampled from a normal distribution with a standard deviation determined by the I/O characteristics of the connected units. We will introduce the Xavier initialization for the FCN, i.e., for the kernels.

**Definition 3.1.33** (Xavier initialization). Let  $\mathbf{K} \in \mathbb{R}^{k_h \times k_w \times z_{in} \times z_{out}}$  be a kernel as in Def. 3.1.12. The initialization

$$\mathbf{K}_{i,j,k,l} \sim \mathcal{N}\left(0, \frac{2}{k_h \cdot k_w \cdot z_{in} + z_{out}}\right) \quad \forall i, j, k, l$$

is called *Xavier initialization*.

This initialization heuristic is designed as compromise between the goals of having the same activation variance as well as gradient variance for all layers. The formula is derived under the assumption of a purely supervised network without any non-linearities. This is obviously not true in most practical cases. Nevertheless, the authors argue that strategies designed for the linear case often perform reasonably well for their non-linear counterparts.

To further reduce the effects of sampling noise (e.g., caused by the choice of the training samples for a minibatch), an exponential moving average is calculated for the model parameters during training. These shadow parameters are not used at all during training, they are just updated. In the end, these weights are used as model parameters for inference. This technique is widely used for image classification purposes.

Although they are just of importance for the briefly introduced ATR system of Chap. 5, we would like to mention two additional concepts which were recently introduced and stabilize as well as improve the ANN training. Dropout [Sri+14] randomly switches off a pre-defined fraction of units in every minibatch step. This should help to prevent the model from overfitting to the training set. Batch normalization [IS15] should accelerate the training and should improve the generalization of the model by normalizing the activations regarding their mean and variance over the current minibatch.

### 3.1.5. Conclusion

This section has described the key aspects of artificial neural networks which are used in this thesis. Besides the classical multilayer perceptron, the fully convolutional network was introduced. It was argued that the FCN is a suitable method to tackle the spatial classification task (or pixel labeling task). For this purpose, an appropriate loss function – the spatial negative log-likelihood – was introduced and motivated. Finally, we have introduced some state-of-the-art methodologies to estimate the model parameters of an FCN given a training set. This process is

referred to as training. A trained FCN is not only able to memorize the training set. Instead, it is also able to generalize to new, previously unseen data which is sampled from the same data distribution as the training set. Besides this ability, the FCN is capable to adapt to entirely new data (different data distribution) via a new training. This is mandatory to handle various different text line intuitions and therefore motivates the usage of FCNs in the context of the text line extraction problem as introduced in Sec. 1.3. See Sec. 4.3 & 4.4 for more details.

## 3.2. Energy Minimization via Graph Cuts

In this section, we will introduce an algorithm to approximately minimize a certain family of energy functions via graph cuts. This section is based on [BVZ01].

**Definition 3.2.1** (labeling problem). Let  $\mathbb{A}$  be a set, and  $l \in \mathbb{N}$  is the number of possible labels. A function  $\Lambda : \mathbb{A} \rightarrow [l]$  is called *labeling* ( $\Lambda$ ). Let  $E$  be a function which maps any labeling to a scalar value called *labeling energy*. The problem defined as

$$\Lambda^* = \arg \min_{\Lambda \text{ is labeling}} E(\Lambda; \mathbb{A})$$

is called *labeling problem*.

Remarkably, any set  $\mathbb{A}$  is allowed, especially any set of pixels. Basically, the labeling problem is an energy minimization problem. *Simulated annealing* [KGV83]; [Čer85] is a general-purpose global energy minimization method which is widely used and could be utilized to solve the labeling problem. There are various different versions of simulated annealing, see [OG12] for a comprehensive review. However, simulated annealing is inefficient in practice, partly, because it changes the labeling of just one element of  $\mathbb{A}$  at each step. Hence, it is very slow for large  $\mathbb{A}$ , see [BVZ01]. To overcome this issue, we will consider only a special form of energy functions and introduce an optimization method based on graph cuts which approximately (but fastly) solves the labeling problem for this kind of energy functions.

### 3.2.1. $\alpha$ - $\beta$ -Swap Energy Minimization

In the following, we consider only energy functions which comprise a data energy term and a smooth energy term. The data energy term penalizes the labeling for each element of  $\mathbb{A}$  separately. On the other hand, the smooth energy term penalizes the difference in the labeling of adjacent elements of  $\mathbb{A}$ . Consequently, a piecewise smooth labeling is desired. This concept is motivated for early vision tasks [VZ99], but is also reasonable for other noisy observations which should be smoothed. We

will introduce a few more termini which are necessary to state the described energy function in a formal way.

**Definition 3.2.2** (neighborhood system, edge, adjacent). Let  $\mathbb{A}$  be a set. We call a subset  $\mathfrak{N} \subset \mathbb{A} \times \mathbb{A}$  *neighborhood system* iff

$$(a, b) \in \mathfrak{N} \Rightarrow a \neq b \wedge (b, a) \in \mathfrak{N} \quad \forall a, b \in \mathbb{A}.$$

Each pair  $(a, b)$  and  $(b, a)$  of elements of  $\mathfrak{N}$  is called *edge* and denoted by  $\mathbf{e}_{a,b}$  (or  $\mathbf{e}_{b,a}$ ).  $a, b$  are *adjacent* iff  $\mathbf{e}_{a,b} \in \mathfrak{N}$  (or  $\mathbf{e}_{b,a} \in \mathfrak{N}$ ).  $\mathbf{e}_{a,b} \setminus a \in \mathbb{A}$  denotes the element  $b$ .

Consequently, edges in a neighborhood system are undirected and self-loops are not allowed. If there is an edge which connects  $a$  and  $b$ , there is just **one** such edge. For the reason of simplicity, we use the notations  $\mathbf{e}_{a,b}$  and  $\mathbf{e}_{b,a}$  for this edge interchangeably. Finally, the energy of Def. 3.2.1 is dependent on a neighborhood system and is defined as

$$E(\Lambda; \mathbb{A}, \mathfrak{N}) := \sum_{a \in \mathbb{A}} D(\Lambda(a); a) + \sum_{\mathbf{e}_{a,b} \in \mathfrak{N}} V(\Lambda(a), \Lambda(b); \mathbf{e}_{a,b}). \quad (3.2.1)$$

with  $D(\cdot; a) : [l] \rightarrow \mathbb{R}$  called *data term* and  $V(\cdot, \cdot; \mathbf{e}_{a,b}) : [l] \times [l] \rightarrow \mathbb{R}_+$  called *smoothing term*. Remarkably, the data term and smoothing term can be (and usually are) entirely different functions for different elements and edges. Examples of these functions are given in Sec. 4.3.2. Here, it is just of importance that  $D$  is arbitrary and  $V$  has to be a *semi-metric*, i.e.,  $V(i, j) = V(j, i) \geq 0$  and  $V(i, j) = 0 \Leftrightarrow i = j$ .

In special cases, such energies can be minimized exactly [BVZ98]; [IG98], but in general the problem is NP-hard [VZ99]. In the following, we introduce a method to approximately minimize the energy which is based on  $\alpha$ - $\beta$ -swaps. We sometimes represent a labeling by its uniquely determined partition (or elements of the partition).

**Definition 3.2.3** (partition). Let  $\mathbb{A}$  be a set. We call a set of subsets  $\{\mathbb{A}^{(1)}, \dots, \mathbb{A}^{(n)}\}$  a *partition* of  $\mathbb{A}$  iff  $\mathbb{A} = \mathbb{A}^{(1)} \cup \dots \cup \mathbb{A}^{(n)} \wedge \mathbb{A}^{(i)} \neq \emptyset \forall i \wedge \mathbb{A}^{(i)} \cap \mathbb{A}^{(j)} = \emptyset \forall i \neq j$ .

Given a labeling  $\Lambda$ , the corresponding partition is defined as

$$\mathbb{A}^{(i)} := \{a \in \mathbb{A} \mid \Lambda(a) = i\} \quad \forall i \in [l], \quad (3.2.2)$$

empty sets are omitted. The notation is simplified by including the empty sets. Hence,  $\mathcal{P} = \{\mathbb{A}^{(1)}, \dots, \mathbb{A}^{(l)}\}$  is the "partition" assigned to labeling  $\Lambda$ . Because of the bijection between labelings and partitions, the terms are used interchangeably.

**Definition 3.2.4** ( $\alpha$ - $\beta$ -swap). Let  $\mathcal{P} = \{\mathbb{A}^{(1)}, \dots, \mathbb{A}^{(l)}\}$  be a partition and  $\alpha, \beta \in [l]$ . A partition  $\mathcal{P}' = \{\tilde{\mathbb{A}}^{(1)}, \dots, \tilde{\mathbb{A}}^{(l)}\}$  is called  $\alpha$ - $\beta$ -swap iff  $i \notin \{\alpha, \beta\} \Rightarrow \mathbb{A}^{(i)} = \tilde{\mathbb{A}}^{(i)}$  holds.

That means, the labeling is just changed for elements which were labeled either with  $\alpha$  or with  $\beta$  in the original labeling. Furthermore, the new label is also in  $\{\alpha, \beta\}$ . It is easy to see that a  $\alpha$ - $\beta$ -swap allows for large moves between two labelings. This is an important property which facilitates a fast convergence of the introduced optimization method. An algorithm which obviously yields a local optimal solution with respect to  $\alpha$ - $\beta$ -swaps is shown in Alg. 2.

---

**Algorithm 2:**  $\alpha$ - $\beta$ -Swap Energy Minimization:  $\Lambda = SWE(E, l, \mathbb{P}, \mathfrak{N})$

---

**input :** energy  $E$  of Eq. (3.2.1), number of labels  $l$ , set of elements  $\mathbb{A}$ ,  
neighborhood system  $\mathfrak{N}$   
**output:** minimized labeling  $\Lambda$

```

1  $\Lambda \leftarrow$  arbitrary labeling
2  $OPT \leftarrow \text{True}$ 
3 while  $OPT$  do
4    $OPT \leftarrow \text{False}$ 
5   for  $\{\alpha, \beta\} \subset [l]$  do
6      $\Lambda^* = \arg \min_{\Lambda'} E(\Lambda'; \mathbb{A}, \mathfrak{N}) \quad \triangleright \Lambda' \text{ results from } \alpha\text{-}\beta\text{-swap for } \Lambda$ 
7     if  $E(\Lambda^*; \mathbb{A}, \mathfrak{N}) < E(\Lambda; \mathbb{A}, \mathfrak{N})$  then
8        $\Lambda \leftarrow \Lambda^*$ 
9        $OPT \leftarrow \text{True}$ 

```

**return :**  $\Lambda$

---

The key part of Alg. 2 is line 6. Now, we demonstrate how graph cuts can be used to efficiently determine the optimal swap move. Graph cuts were extensively studied and the optimal solution can be found in low-order polynomial complexity [AMO14]. Boykov et al. [BK04] introduce a max-flow algorithm which has a nearly linear running time in practice. In this thesis, we describe the graph cut part only. For a detailed description of suitable methods to solve this problem, we refer to the above-mentioned literature. In the rest of this section, we will introduce the graph cut problem. Afterwards, it is proven that the finding of an optimal swap move is equivalent to the finding of a minimum graph cut for a certain graph.

### 3.2.2. Optimal $\alpha$ - $\beta$ -Swap via Graph Cuts

First, we introduce the necessary termini from the field of graph theory.

**Definition 3.2.5** (graph). Let  $\mathbb{A}$  be a set.  $\mathfrak{N}$  is a neighboring system (cf. Def. 3.2.2) over  $\mathbb{A}$ .  $(\mathbb{A}, \mathfrak{N})$  is called *graph*. The elements of  $\mathbb{A}$  are called *vertex*.

**Definition 3.2.6** (connected). Let  $(\mathbb{A}, \mathfrak{N})$  be a graph. Two vertices  $a, b$  are *connected* iff

$$\exists a^{(0)}, \dots, a^{(N)} \in \mathbb{A} : a^{(0)} = a \wedge a^{(N)} = b \wedge e_{a^{(i)}, a^{(i+1)}} \in \mathfrak{N} \quad (0 \leq i \leq N-1)$$



holds. The chain of edges connecting  $a$  and  $b$  is called *path*.

**Definition 3.2.7** (graph cut). Let  $(\mathbb{A}^{(\alpha\beta)}, \mathfrak{N})$  be a graph with two distinct vertices which are assigned to be *terminals*, i.e.,  $\mathbb{A}^{(\alpha\beta)} = \mathbb{A} \cup \{t^{(\alpha)}, t^{(\beta)}\}$ . A subset  $\mathfrak{N}_C \subset \mathfrak{N}$  is called *minimal graph cut* iff the terminals are separated, i.e., not connected, in the graph  $(\mathbb{A}, \mathfrak{N} \setminus \mathfrak{N}_C)$  and no proper subset of  $\mathfrak{N}_C$  separates the terminals.

A graph becomes a *weighted graph* if there is a *weight function*  $w : \mathfrak{N} \rightarrow \mathbb{R}_+$  assigning a (non-negative) weight to each edge. The minimum graph cut problem can now be stated as follows.

**Definition 3.2.8** (minimum cut problem). Let  $(\mathbb{A}^{(\alpha\beta)}, \mathfrak{N}, w)$  define a weighted graph. The *minimum cut problem* is defined as

$$\mathfrak{N}_C^* = \arg \min_{\mathfrak{N}_C \text{ is graph cut}} \sum_{e_{p,q} \in \mathfrak{N}_C} w(e_{p,q}).$$

As mentioned above, there are efficient algorithms to solve this problem, but this is beyond the scope of this work.

For the rest of this section,  $\mathbb{A}$  is the set of elements to be labeled,  $l$  is the number of labels,  $\mathfrak{N}$  is a neighborhood system over  $\mathbb{A}$ , and  $E, D, V$  are of Eq. (3.2.1). Let  $\mathcal{P}$  be the current partition (labeling  $\Lambda$ ) and  $\alpha, \beta \in [l]$  are two labels for which the optimal swap has to be determined (cf. line 6 of Alg. 2). Now, we present the construction of the graph for which the minimum cut yields the optimal swap move.

Let  $\mathbb{A}^{(\alpha)}, \mathbb{A}^{(\beta)}$  be the subsets of  $\mathbb{A}$  which are labeled with  $\alpha$  and  $\beta$ , respectively, cf. Eq. (3.2.2). Let  $t^{(\alpha)}$  and  $t^{(\beta)}$  denote the two terminals. Note, these are additional vertices and they are **not** in  $\mathbb{A}$ . The set  $\mathbb{A}^{(\alpha\beta)} := \mathbb{A}^{(\alpha)} \cup \mathbb{A}^{(\beta)} \cup \{t^{(\alpha)}, t^{(\beta)}\}$  constitutes the set of vertices of the graph to be constructed. The set of edges is defined as

$$\mathfrak{N}^{(\alpha\beta)} := \{e_{a,b} \in \mathfrak{N} \mid a, b \in \mathbb{A}^{(\alpha)} \cup \mathbb{A}^{(\beta)}\} \cup \bigcup_{a \in \mathbb{A}^{(\alpha)} \cup \mathbb{A}^{(\beta)}} \{e_{a,t^{(\alpha)}}, e_{a,t^{(\beta)}}\}.$$

In other words, the new set of vertices contains all elements which are labeled with  $\alpha$  or  $\beta$  and the two terminals. The new set of edges comprises all edges of  $\mathfrak{N}$  connecting elements which are labeled with  $\alpha$  or  $\beta$ . Additionally, edges connecting each of the  $\alpha, \beta$  labeled elements with the terminals are added, see Fig. 3.1a. Let  $\mathfrak{N}(a)$  denote the set of elements in  $\mathbb{A}$  which are adjacent to  $a$  and not labeled with  $\alpha$  nor with  $\beta$ . Finally, the weight function  $w_{\alpha\beta} : \mathfrak{N}^{(\alpha\beta)} \rightarrow \mathbb{R}_+$  is defined as follows

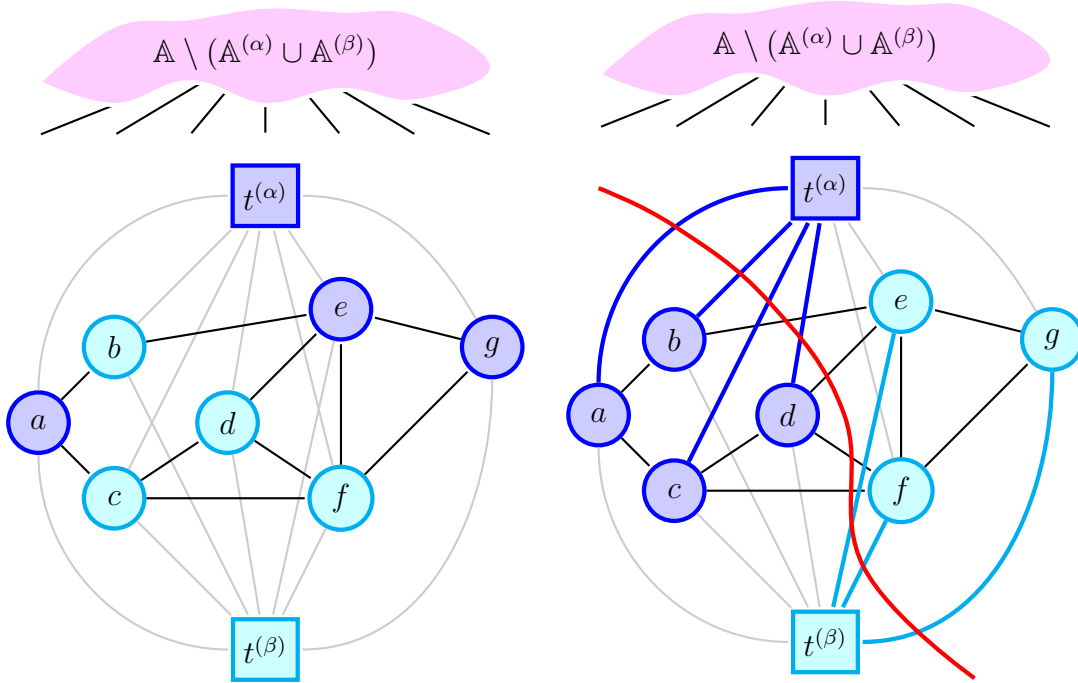
$$w(e_{a,b}) := \begin{cases} V(\alpha, \beta; e_{a,b}) & , \text{ if } a, b \in \mathbb{A}^{(\alpha)} \cup \mathbb{A}^{(\beta)} \\ D(\alpha; a) + \sum_{c \in \mathfrak{N}(a)} V(\alpha, \Lambda(c); e_{a,c}) & , \text{ if } b = t^{(\alpha)} \\ D(\beta; a) + \sum_{c \in \mathfrak{N}(a)} V(\beta, \Lambda(c); e_{a,c}) & , \text{ if } b = t^{(\beta)} \end{cases}. \quad (3.2.3)$$

Let  $\mathfrak{N}_C^{(\alpha\beta)}$  be a minimum cut for the constructed weighted graph.

**Lemma 3.2.9.**  $\mathfrak{N}_C^{(\alpha\beta)}$  includes either  $e_{a,t^{(\alpha)}}$  or  $e_{a,t^{(\beta)}}$  for all  $a \in \mathbb{A}^{(\alpha)} \cup \mathbb{A}^{(\beta)}$ .

*Proof.* Assume there is an  $a \in \mathbb{A}$  with  $e_{a,t^{(\alpha)}}, e_{a,t^{(\beta)}} \notin \mathfrak{N}_C^{(\alpha\beta)}$ , then the two terminals are connected via  $a$ .  $\nmid$

On the other hand, assume there is an  $a \in \mathbb{A}$  with  $e_{a,t^{(\alpha)}}, e_{a,t^{(\beta)}} \in \mathfrak{N}_C^{(\alpha\beta)}$ . Let  $\mathbb{A}^{(a)}$  denote the set of vertices which are connected to  $a$  in  $(\mathbb{A}, \mathfrak{N} \setminus \mathfrak{N}_C^{(\alpha\beta)})$ . The elements of  $\mathbb{A}^{(a)}$  are either all connected to  $t^{(\alpha)}$  or to  $t^{(\beta)}$ , otherwise there would be a path connecting  $t^{(\alpha)}$  and  $t^{(\beta)}$  via  $a$ . Assume  $\mathbb{A}^{(a)}$  is connected to  $t^{(\alpha)}$  (w.l.o.g.) then  $\mathfrak{N}_C^{(\alpha\beta)} \setminus e_{a,t^{(\alpha)}}$  is a proper subset of  $\mathfrak{N}_C^{(\alpha\beta)}$  and separates the two terminals.  $\nmid$   $\square$



(a) Shown is the initial labeling for  $\alpha$  and  $\beta$ :  $\mathbb{A}^{(\alpha)} = \{a, e, g\}$  and  $\mathbb{A}^{(\beta)} = \{b, c, d, f\}$  and the resulting graph for which the minimum cut is estimated. (b) The minimum cut is depicted as red line. The labeling after the optimal  $\alpha$ - $\beta$ -swap differs just in  $\mathbb{A}^{(\alpha)} = \{a, b, c, d\}$  and  $\mathbb{A}^{(\beta)} = \{e, f, g\}$  from the initial labeling.

**Figure 3.1.:  $\alpha$ - $\beta$ -swap optimization via graph cut** – Shown is the optimal  $\alpha$ - $\beta$ -swap estimation utilizing the minimum cut. In the initial labeling,  $\mathbb{A}^{(\alpha)} \cup \mathbb{A}^{(\beta)} = \{a, b, c, d, e, f, g\}$  are labeled with  $\alpha$  (blue) or  $\beta$  (cyan). The set of elements which is labeled with  $[l] \setminus \{\alpha, \beta\}$  is shown in magenta. The neighborhood system  $\mathfrak{N}$  is represented by the black lines. The additional edges which connect the terminals to  $\mathbb{A}^{(\alpha)} \cup \mathbb{A}^{(\beta)}$  are shown in gray. The edges leaving the magenta set are solely of interest for the weight function, cf. Eq. (3.2.3), but they are **not** part of the graph for which the minimum cut is estimated.

Consequently, there is a generic labeling

$$\tilde{\Lambda}(a) := \begin{cases} \alpha & , \text{ if } \mathbf{e}_{a,t^{(\alpha)}} \in \mathfrak{N}_C^{(\alpha\beta)} \\ \beta & , \text{ if } \mathbf{e}_{a,t^{(\beta)}} \in \mathfrak{N}_C^{(\alpha\beta)} \\ \Lambda(a) & , \text{ else} \end{cases} \quad (3.2.4)$$

which is well-defined for all  $a \in \mathbb{A}$  and determined by the minimum cut.

**Theorem 3.2.10.**  $\tilde{\Lambda}$  is the optimal  $\alpha$ - $\beta$ -swap from  $\Lambda$ .

*Proof.* First, there is a one-to-one correspondence between cuts on the constructed graph and labelings which are one  $\alpha$ - $\beta$ -swap from  $\Lambda$ . The first direction of this equivalence follows from Lem. 3.2.9 and Eq. (3.2.4). The second direction follows, because each labeling uniquely determines the edges of  $\mathfrak{N}$  which connect differently  $\alpha$ ,  $\beta$ -labeled vertices as well as edges connecting the vertices to the terminals of the appropriate label.

Now, we calculate the weight of a cut and show that it is equal to the energy of the resulting labeling plus a constant which is not influenced by  $\alpha$ - $\beta$ -swaps. The weight of the cut comprises the summation over the weights of the different types of edges,

$$\begin{aligned} w(\mathfrak{N}_C^{(\alpha\beta)}) &= \sum_{\mathbf{e}_{a,t^{(\alpha)}} \in \mathfrak{N}_C^{(\alpha\beta)}} w(\mathbf{e}_{a,t^{(\alpha)}}) + \sum_{\mathbf{e}_{a,t^{(\beta)}} \in \mathfrak{N}_C^{(\alpha\beta)}} w(\mathbf{e}_{a,t^{(\beta)}}) + \sum_{\substack{\mathbf{e}_{a,b} \in \mathfrak{N}_C^{(\alpha\beta)}, \\ a,b \notin \{t^{(\alpha)}, t^{(\beta)}\}}} w(\mathbf{e}_{a,b}) \\ &= \sum_{\mathbf{e}_{a,t^{(\alpha)}} \in \mathfrak{N}_C^{(\alpha\beta)}} \left( D(\alpha; a) + \sum_{c \in \mathfrak{N}(a)} V(\alpha, \Lambda(c); \mathbf{e}_{a,c}) \right) \\ &\quad + \sum_{\mathbf{e}_{a,t^{(\beta)}} \in \mathfrak{N}_C^{(\alpha\beta)}} \left( D(\beta; a) + \sum_{c \in \mathfrak{N}(a)} V(\beta, \Lambda(c); \mathbf{e}_{a,c}) \right) \\ &\quad + \sum_{\substack{\mathbf{e}_{a,b} \in \mathfrak{N}_C^{(\alpha\beta)}, \\ a,b \notin \{t^{(\alpha)}, t^{(\beta)}\}}} V(\alpha, \beta; \mathbf{e}_{a,b}) \\ &= \sum_{a \in \mathbb{A}^{(\alpha)} \cup \mathbb{A}^{(\beta)}} D(\tilde{\Lambda}(a); a) + \sum_{\substack{\mathbf{e}_{a,b} \in \mathfrak{N}, \\ \{a,b\} \cap (\mathbb{A}^{(\alpha)} \cup \mathbb{A}^{(\beta)}) \neq \emptyset}} V(\tilde{\Lambda}(a), \tilde{\Lambda}(b); \mathbf{e}_{a,b}). \end{aligned}$$

The last equality holds, because  $V(\alpha, \alpha; \mathbf{e}_{a,b}) = V(\beta, \beta; \mathbf{e}_{a,b}) = 0$ . Finally,

$$E(\tilde{\Lambda}) = w(\mathfrak{N}_C^{(\alpha\beta)}) + K$$

with

$$K = \sum_{a \notin \mathbb{A}^{(\alpha)} \cup \mathbb{A}^{(\beta)}} D(\tilde{\Lambda}(a); a) + \sum_{\substack{\mathbf{e}_{a,b} \in \mathfrak{N}, \\ \{a,b\} \cap (\mathbb{A}^{(\alpha)} \cup \mathbb{A}^{(\beta)}) = \emptyset}} V(\tilde{\Lambda}(a), \tilde{\Lambda}(b); \mathbf{e}_{a,b})$$

holds. Consequently, the weight of the cut is equal to the energy of the resulting labeling plus a constant which is independent to  $\alpha, \beta$ -labeled elements. Together with the one-to-one correspondence between cuts and labelings the theorem is proven.  $\square$

**Remark 3.2.11.** Of course, a local minimum with respect to swap moves can be arbitrarily far away from the global optimum. Boykov et al. [BVZ01] propose another energy minimization method which is based on expansion moves. For a given label  $\alpha$ , an expansion move can relabel any element with the label  $\alpha$ . This method has two major advantages over the  $\alpha$ - $\beta$ -swap algorithm. First, the  $\alpha$ - $\beta$ -swap algorithm is quadratic in the choice of the labels to swap, cf. Alg. 2 line 5. Here, the  $\alpha$ -expansion algorithm is linear. Furthermore, it is proven in [VZ99] that a local minimum with respect to expansion moves ( $\tilde{\Lambda}$ ) is within a certain range of the global minimum ( $\Lambda^*$ )

$$E(\tilde{\Lambda}) \leq 2 \cdot c \cdot E(\Lambda^*)$$

with

$$c = \max_{e_{a,b} \in \mathfrak{N}} \frac{\max_{i \neq j \in [l]} V(i, j; e_{a,b})}{\min_{i \neq j \in [l]} V(i, j; e_{a,b})}.$$

The major drawback is that  $V$  has to be a metric, i.e., the triangle inequality has to hold. This is not the case in our scenarios. Consequently, the  $\alpha$ - $\beta$ -swap algorithm is used throughout this thesis. It has to be mentioned that the  $\alpha$ - $\beta$ -swap algorithm yields results which are satisfying in all our scenarios.

### 3.2.3. Conclusion

This section has introduced an energy minimization framework based on graph cuts which will be used to estimate certain text line characteristics which are used for the baseline estimation in Sec. 4.3. The energy function comprises a data term and a smoothing term. The smoothing term penalizes differences in the labeling of elements which are assigned to be adjacent. We will utilize this to encode the expectation that some characteristics of text lines tend to change smoothly for nearby text lines, see Sec. 4.3.



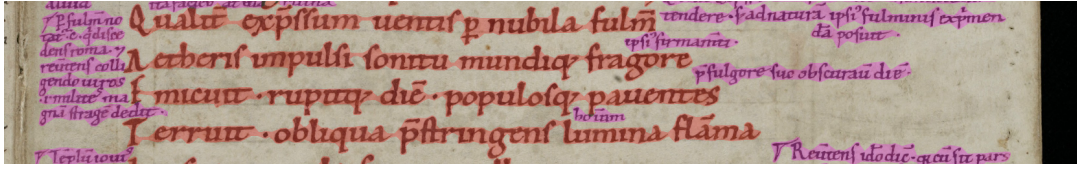
## 4. Baseline Detection

The text line extraction problem as introduced in Sec. 1.3 depends on a text line intuition (TLI). Because the text line extraction is just the first step in the information retrieval pipeline on scanned documents this TLI is usually the result of an (human) operator combined with the demands of the ATR/KWS modules which process the extracted text lines. Basically, these TLIs differ in three major questions:

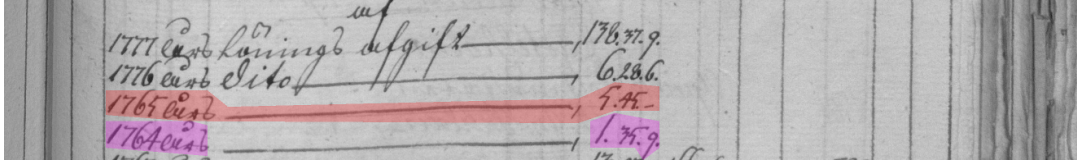
- (a) *What is a text line?* E.g., are we just interested in main body text, or also in marginalia text as well as comments? (cf. Fig. 4.1a)
- (b) *How does the page layout look like?* E.g., where does a text line begin and where does it end? (cf. Fig. 4.1b)
- (c) *How should a text line be represented?* E.g., by a bounding box or by a more complex polygon? (cf. Fig. 4.1c)

The first two questions depend on the (human) operator, the third question depends on the respective ATR/KWS system. Hence, it is meaningful to decouple the text line extraction problem into two subproblems. First, the text lines should be detected and encoded by a meaningful representation, i.e., questions (a) & (b) should be "answered". Second, the detected text lines should be extracted, i.e., question (c) should be answered. It is easy to see that this results in a flexible text line extraction method. The first subproblem tries to satisfy the demands of the (human) operator (first two questions), the second subproblem could be adapted to the ATR/KWS system which should be used to transform the pixel information into textual information. But how does a meaningful representation of a detected text line look like? Murdock et al. [Mur+15] encoded each text line by its origin point. Roughly speaking, this is the left-bottom corner of a text line. In our opinion, this representation does not encode sufficient information, especially questions (b) & (c) are not answered at all. To overcome this limitation we decided to encode the detected text lines by their baselines. This representation has the advantage that it encodes the general layout characteristics of the underlying TLI, i.e., separation issues like where a text line begins and where it ends. Thus, question (b) is answered. Given a baseline it is comparatively easy to extract the text line in a suitable representation, see Chap. 5. Finally, one can set up a specific "baseline to text line" module if a certain ATR/KWS necessitates a special text line representation.

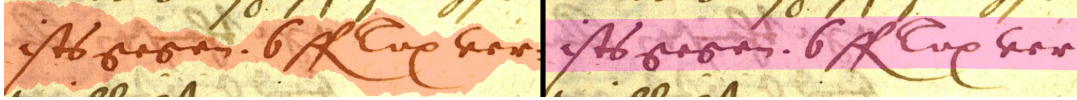
After briefly introducing the baseline detection problem, a newly developed similarity score for baselines is described. The rest of the chapter is devoted to the proposed baseline detection method.



- (a) Depicted is a snippet of an image of the DIVA-HisDB [Sim+16]. Besides the main body text (red), there are also comments (magenta). The TLI of [Sim+17] is limited to the main body text. Of course, TLIs which are interested in the entire text are also reasonable.



- (b) Depicted is a snippet of an image of the cBad test set [Grü+18a]. Two different TLIs (red and magenta) are outlined which differ in the understanding where a text line starts and where it ends end.



- (c) Depicted is a snippet of an image of the Bozen test set [Sán+16]. Two different representations of the same text line are shown. The left one is detailed and takes care of all ascenders and descenders. The right one is simpler and cuts some descenders, see Chap. 5.

Figure 4.1.: **Differences in the text line intuitions** – This figure demonstrates the major issues in which TLIs typically differ.

## 4.1. Problem Statement

We will briefly introduce the problem of baseline detection. Because its formulation is related to Sec. 1.3 and builds upon it (and its notation), we will often refer to Sec. 1.3 and will not state definitions which are just slightly different to already introduced ones.

Within this work we follow the definition of a baseline given in [Die+17b]:

**Definition 4.1.1** (baseline). A *baseline* is defined in the typographical sense as the virtual line where most characters rest upon and descenders extend below.

To encode baselines in a mathematical way, we introduce the polygonal chain.

**Definition 4.1.2** (polygonal chain). A *polygonal chain* of length  $n \in \mathbb{N}$  is an  $n$ -tuple of pixels

$$\mathcal{P} = (\mathbf{p}^{(1)}, \mathbf{p}^{(2)}, \dots, \mathbf{p}^{(n)}). \quad (4.1.1)$$

A polygonal chain is said to be *closed* iff  $\mathbf{p}^{(1)} = \mathbf{p}^{(n)}$  holds. Two pixels  $\mathbf{p}^{(i)}, \mathbf{p}^{(j)}$  of a polygonal chain are called *adjacent* iff  $|i - j| = 1$ .

For the reason of simplicity, we write  $\mathbf{p} \in \mathcal{P}$  if a pixel belongs to the polygonal chain (which is not a set but a tuple).

Consequently, each baseline can be represented by a polygonal chain. Of course, the mapping from baselines to polygonal chains is not well-defined. There are various different polygonal chains which could be assigned to a baseline, see Fig. 4.2. This has to be taken into account if measuring the accuracy of a detected baseline. The choice of the representative should not negatively affect the accuracy. Since we assume that each baseline is represented by a polygonal chain, we sometimes mix the termini baseline and polygonal chain. The equivalent to the text line space ( $\mathfrak{P}_{TL}$ ) is the baseline space.

**Definition 4.1.3** (baseline space). The infinite set of all polygonal chains is called *baseline space* ( $\mathfrak{P}_{BL}$ ).

We will not introduce the *baseline intuition* (*BLI*)  $\tilde{\iota}_I : \mathfrak{I} \rightarrow 2^{\mathfrak{P}_{BL}}$  (cf. Def. 1.3.7), (*baseline*) *ground truth* ( $\tilde{\iota}_I(I)$ ) (cf. Def. 1.3.7), *baseline detector* (*BLD*)  $\iota_D : \mathfrak{I} \rightarrow 2^{\mathfrak{P}_{BL}}$  (cf. 1.3.8), *baseline hypothesis* ( $\iota_D(I)$ ) (cf. 1.3.8), (*baseline*) *similarity score*  $\langle \cdot, \cdot \rangle_{BL} : 2^{\mathfrak{P}_{BL}} \times 2^{\mathfrak{P}_{BL}} \rightarrow [0, 1]$  (cf. 1.3.9), (*baseline*) *test set*  $\mathfrak{T}_{\tilde{\iota}_I}$  (cf. Eq. (1.3.2)), (*baseline*) *train set*  $\mathfrak{T}'_{\tilde{\iota}_I}$  (cf. Eq. (1.3.3)), and (*baseline*) *test score*  $\tilde{\tau}_{test}$  (cf. Def. 1.3.10) thoroughly and refer to Sec. 1.3. These are just the generic versions for baselines instead of text lines. Hence, also the baseline detection problem can be stated similar to the text line extraction problem.

**Baseline Detection Problem** Assume there is a baseline intuition along with a (*baseline*) test set  $\mathfrak{T}_{\tilde{\iota}_I}$  and a (*baseline*) train set  $\mathfrak{T}'_{\tilde{\iota}_I}$  which were generated by means of the BLI. Design a baseline detector ( $\iota_D^*$ ) which utilizes just  $\mathfrak{T}'_{\tilde{\iota}_I}$  and maximizes the test score  $\tilde{\tau}_{test}$  on  $\mathfrak{T}_{\tilde{\iota}_I}$ .

The demand for an easy adaptation to different TLIs of Sec. 1.3 is also valid for different BLIs. As a result of the argumentation in the introduction to this chapter, the ability to adapt to different BLIs is equivalent to the ability to adapt to different TLIs (regarding questions (a) & (b)). Hence, this ability constitutes a central demand on a BLD.

## 4.2. Similarity Score

This section describes the evaluation scheme introduced in [Grü+18a] in a notation adapted to this work. This evaluation scheme was already used in two international competitions dealing with baseline detection in historical documents [Sim+17]; [Die+17b]. These competitions were organized in conjunction with the 14th IAPR International Conference on Document Analysis and Recognition. Based upon this scheme, the accuracy of a BLD is measured by means of the F-value.



Finally, the F-value will constitute the (baseline) similarity score  $\langle \cdot, \cdot \rangle_{BL} : 2\mathfrak{P}_{BL} \times 2\mathfrak{P}_{BL} \rightarrow [0, 1]$ .

### 4.2.1. Motivation & Requirements

Since the widely-used similarity scores for text line extraction [Li+08]; [GSL11]; [Sim+17] rely on surrounding polygons and use area (or foreground pixel) based methods to calculate the accuracy of text line segmentation results, there is a need for a similarity score suitable for baselines. To our knowledge there was no such similarity score before the one introduced in [Grü+18a].

There are special requirements for a similarity score developed to measure the accuracy of a BLD:

- (I) The similarity score should be invariant to small differences between ground truth (GT) and hypothesis (HYP). Def. 4.1.1 as well as the representation of a baseline by a polygonal chain allow for a certain margin of "correct" baselines. Furthermore, slightly different baselines potentially lead to the same ATR/KWS accuracy. Hence, there is not one unique and correct HYP for an image (or even for a GT), this has to be taken into account.
- (II) The similarity score should be able to handle skewed and oriented text lines.
- (III) The similarity score should only rely on the baseline GT and baseline HYP, not on a reading order nor on any image information, e.g., on a binarization of the image.

Besides these requirements which have to be fulfilled in general, there are two characteristics of the baseline HYP which also should be evaluated:

- (i) It has to be evaluated how reliable the text is detected – ignoring layout issues. The value reflecting this is called R-value since it has similar properties as the well-known *recall* value, see [Pow11].
- (ii) It has to be evaluated how reliable the structure of the text lines (layout) of the document is detected. The value reflecting this is called P-value since it has similar properties as the well-known *precision* value, see [Pow11].

These values are useful to analyze a BLD concerning its strengths and weaknesses. Furthermore, they are related to the expected/possible performance of the subsequent ATR/KWS modules in the information retrieval pipeline. Finally, the R-value and the P-value should be combined to constitute the similarity score. We refer to this value as F-value. The algorithm calculating all three values, i.e., R-value, P-value, and F-value, is referred to as evaluation scheme.

In the following, we will introduce an evaluation scheme which meets all the above stated requirements. The proposed evaluation scheme is implemented in Java and available as a standalone command line tool. It is licensed under LGPLv3 and

publicly available<sup>1</sup>.

### 4.2.2. Evaluation Scheme

Now, the calculation of the R-value and the P-value for the HYP and GT for a single image  $I \in \mathfrak{I}$  is described. The baseline GT  $\tilde{\iota}_I(I)$  for image  $I$  and the baseline HYP  $\iota_D(I)$  for image  $I$  are denoted by

$$\tilde{\iota}_I(I) = \mathbb{G} = \{\mathcal{G}^{(1)}, \dots, \mathcal{G}^{(M)}\} \in 2^{\mathfrak{P}_{BL}}, \quad (4.2.1)$$

$$\iota_D(I) = \mathbb{H} = \{\mathcal{H}^{(1)}, \dots, \mathcal{H}^{(K)}\} \in 2^{\mathfrak{P}_{BL}}. \quad (4.2.2)$$

See Fig. 4.4a for exemplary sets  $\mathbb{G}$  and  $\mathbb{H}$ . The calculation of the R-value and the P-value for the two sets  $\mathbb{G}$  and  $\mathbb{H}$  is described in the following.

### Polygonal Chain Normalization

In a first step, each polygonal chain is normalized such that two adjacent pixels are (roughly) of predefined distance. This is done utilizing line segments.

**Definition 4.2.1** (line segment). Let  $\mathbf{p}, \mathbf{q}$  be two pixels. The function  $\lambda(\cdot; \mathbf{p}, \mathbf{q}) : [0, 1] \rightarrow \mathbb{R}^2$  defined as

$$\lambda(\tau; \mathbf{p}, \mathbf{q}) := \mathbf{p} + \tau \cdot (\mathbf{q} - \mathbf{p}) \quad (4.2.3)$$

is called *line segment*  $(\lambda(\cdot; \mathbf{p}, \mathbf{q}))$  connecting  $\mathbf{p}$  and  $\mathbf{q}$ .

A polygonal chain  $\mathcal{P}$  is normalized by  $\tilde{\mathcal{P}} := \text{NORM}(\mathcal{P}, d)$ , see Alg. 3.

Hence, adjacent pixels of the normalized polygonal chains have a maximal distance  $d$  (except for rounding errors and border effects). The resulting sets of normalized chains are  $\tilde{\mathbb{G}}$  and  $\tilde{\mathbb{H}}$ . For a better readability, we omit the tilde. Consequently,  $\mathbb{G}$  and  $\mathbb{P}$  denote the sets of normalized polygonal chains in the following. The normalization step is mandatory to compensate the effect of the choice of the representative of a baseline, see Fig. 4.2.

### Tolerance Value Calculation

As mentioned above, the evaluation scheme should not penalize HYP baselines which are "slightly" different to the GT baselines. Thus, some kind of tolerance area around each GT polygonal chain is required. Image (and baseline) dependent tolerance values are calculated because various resolutions and layout scenarios could be present

<sup>1</sup><https://github.com/Transkribus/TranskribusBaseLineEvaluationScheme>

---

**Algorithm 3:** Polygonal Chain Normalization:  $\tilde{\mathcal{P}} = \text{NORM}(\mathcal{P}, d)$ 


---

**input** : polygonal chain  $\mathcal{P} = (\mathbf{p}^{(1)}, \dots, \mathbf{p}^{(n)})$ , distance value  $d$   
**output**: normalized polygonal chain  $\tilde{\mathcal{P}}$

```

1  $\tilde{\mathcal{P}} \leftarrow \emptyset$ 
2 for  $i = 1$  to  $n - 1$  do
3    $l \leftarrow \|\mathbf{p}^{(i)} - \mathbf{p}^{(i+1)}\|_2$ 
4    $s \leftarrow \lceil l/d \rceil$ 
5    $\tilde{\mathcal{P}} \leftarrow \tilde{\mathcal{P}} \cup \mathbf{p}^{(i)}$ 
6   for  $j = 1$  to  $s - 1$  do
7      $\tilde{\mathcal{P}} \leftarrow \tilde{\mathcal{P}} \cup \lambda(j/s; \mathbf{p}^{(i)}, \mathbf{p}^{(i+1)}) \triangleright \lambda$  mapped to a pixel (Rem. 1.3.3)
8  $\tilde{\mathcal{P}} \leftarrow \tilde{\mathcal{P}} \cup \mathbf{p}^{(n)}$ 
return :  $\tilde{\mathcal{P}}$ 

```

---

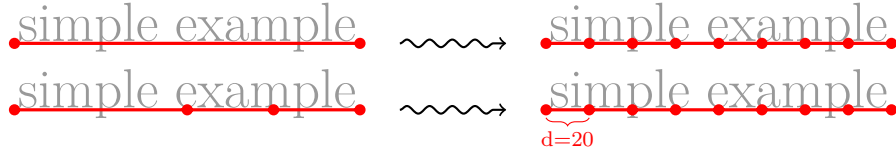


Figure 4.2.: **Polygonal chain normalization** – Depicted is the normalization of two different representatives of the same baseline. Only two points of the lower left baseline have a correspondence in the one above. The two intermediate points do not contribute any additional information (straight line). Hence, the normalizations result in two equivalent polygonal chains.

within a test set of images. A single pre-defined tolerance value can hardly cover all these scenarios in a satisfying fashion. Hence, for each polygonal chain  $\mathcal{G} \in \mathbb{G}$  a *tolerance value*  $t_{\mathcal{G}}$  is calculated. Remarkably, the term "slightly" is determined by the respective tolerance value. The calculation of the tolerance value for a certain GT baseline is based on the distance to the "closest" (this will be clarified below) other GT baseline. The calculation of this distance and the resulting tolerance value is described in the following.

First, the text orientation of the baselines is roughly estimated by means of linear regression. The resulting orientation is encoded in an orientation vector.

**Definition 4.2.2** (polygonal chain orientation). Let  $\mathcal{P} \in \mathfrak{P}_{BL}$  be a polygonal chain. The slope ( $\omega_{\mathcal{P}} \in [0, \pi)$ ) of the line solving the linear regression problem for the pixels of  $\mathcal{P}$  is called *polygonal chain orientation* ( $\omega_{\mathcal{P}}$ ).

**Definition 4.2.3** (orientation vector). Let  $\alpha \in [0, \pi)$  be an angle. The vector of

length one defined as

$$\mathbf{o}^{(\alpha)} := (-\sin(\alpha), \cos(\alpha))^T \in \mathbb{R}^2 \quad (4.2.4)$$

is called *orientation vector* ( $\mathbf{o}^{(\alpha)}$ ).

For a better understanding of Eq. (4.2.4), remember that the first dimension constitutes the  $y$ -coordinate and the  $y$ -axis is inverted, cf. Rem. 1.3.2. The concept of off-text distance and in-text distance is introduced and utilized to determine the line-dependent tolerance value.

**Definition 4.2.4** (off-text distance, in-text distance). Let  $\mathbf{o}^{(\alpha)}$  be an orientation vector. For two pixels  $\mathbf{p}, \mathbf{q}$ , the length of the component of  $(\mathbf{p} - \mathbf{q}) \in \mathbb{R}^2$  which is orthogonal to  $\mathbf{o}^{(\alpha)}$  is called *off-text distance* of  $\mathbf{p}$  and  $\mathbf{q}$ . It is denoted by  $d_{off}(\mathbf{p}, \mathbf{q}; \alpha)$ . The length of the component of  $(\mathbf{p} - \mathbf{q}) \in \mathbb{R}^2$  which has the same direction as  $\mathbf{o}^{(\alpha)}$  is called *in-text distance* and is denoted by  $d_{in}(\mathbf{p}, \mathbf{q}; \alpha)$ .

The concept of off-text direction and in-text direction is visualized in Fig. 4.3.

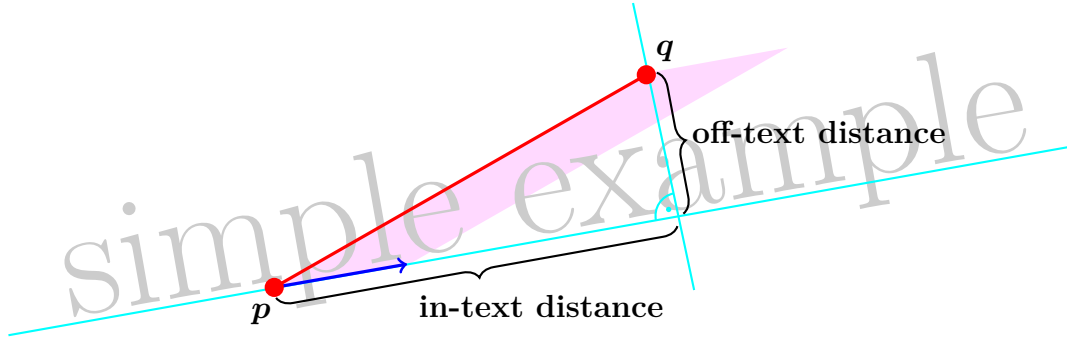


Figure 4.3.: **Off-text distance and in-text distance for two pixels** – The concept of off-text distance and in-text distance for two pixels  $\mathbf{p}, \mathbf{q}$  and an orientation vector (blue arrow) is displayed. The orientation vector encodes the text direction. The off-text distance is the component of  $(\mathbf{p} - \mathbf{q})$  which is orthogonal to the text direction. The in-text distance is the component which has the text orientation. The magenta parallelogram is utilized for an efficient calculation of the distances, see main text.

**Lemma 4.2.5.** Let  $\mathbf{o}^{(\alpha)}$  be an orientation vector and  $\mathbf{p}, \mathbf{q}$  are two pixels. The off-text distance can be efficiently calculated by

$$d_{off}(\mathbf{p}, \mathbf{q}; \alpha) = \left| -(\mathbf{p}_2 - \mathbf{q}_2)\mathbf{o}^{(\alpha)}_1 + (\mathbf{p}_1 - \mathbf{q}_1)\mathbf{o}^{(\alpha)}_2 \right|. \quad (4.2.5)$$

The in-text distance can be calculated by

$$d_{in}(\mathbf{p}, \mathbf{q}; \alpha) = \left| (\mathbf{p} - \mathbf{q})^T \mathbf{o}^{(\alpha)} \right|. \quad (4.2.6)$$

*Proof.* Eq. (4.2.5) follows from some basic geometric considerations. The area of the magenta parallelogram in Fig. 4.3 equals the off-text distance, because the orientation vector has length 1. The assertion follows from the transformation matrix

$$T = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

(from computer vision coordinates to classical coordinates) and the cross product

$$\left( T \cdot \begin{pmatrix} \mathbf{p}_1 - \mathbf{q}_1 \\ \mathbf{p}_2 - \mathbf{q}_2 \\ 0 \end{pmatrix} \right) \times \left( T \cdot \begin{pmatrix} \mathbf{o}^{(\alpha)}_1 \\ \mathbf{o}^{(\alpha)}_2 \\ 0 \end{pmatrix} \right) = \begin{pmatrix} 0 \\ 0 \\ -(\mathbf{p}_2 - \mathbf{q}_2)\mathbf{o}^{(\alpha)}_1 + (\mathbf{p}_1 - \mathbf{q}_1)\mathbf{o}^{(\alpha)}_2 \end{pmatrix},$$

and the fact that its length is equal to the parallelogram's area.

The projection of  $(\mathbf{p} - \mathbf{q})$  on  $\mathbf{o}^{(\alpha)}$  yields

$$\frac{\mathbf{o}^{(\alpha)T} \cdot T^T \cdot T \cdot (\mathbf{p} - \mathbf{q})}{\mathbf{o}^{(\alpha)T} \cdot T^T \cdot T \cdot \mathbf{o}^{(\alpha)}} \cdot \mathbf{o}^{(\alpha)}.$$

Eq. (4.2.6) follows, because  $T$  is orthogonal and  $\mathbf{o}^{(\alpha)}$  has length 1. Consequently,

$$\left| \frac{\mathbf{o}^{(\alpha)T} \cdot (\mathbf{p} - \mathbf{q})}{\mathbf{o}^{(\alpha)T} \cdot \mathbf{o}^{(\alpha)}} \cdot \mathbf{o}^{(\alpha)} \right| = \left| \mathbf{o}^{(\alpha)T} \cdot (\mathbf{p} - \mathbf{q}) \right| \cdot \left\| \mathbf{o}^{(\alpha)} \right\|_2 = \left| (\mathbf{p} - \mathbf{q})^T \cdot \mathbf{o}^{(\alpha)} \right|$$

holds. □

Let  $\mathcal{G} \in \mathbb{G}$  be any normalized GT polygonal chain. Let  $\mathbb{X}$  be the set of all pixels of the chains of  $\mathbb{G} \setminus \mathcal{G}$ , the set  $\mathbb{X}_{\mathcal{G}} \subset \mathbb{X}$  is calculated such that for any  $\mathbf{p} \in \mathbb{X}_{\mathcal{G}}$  there are at least two pixels  $\mathbf{p}^{(1)}, \mathbf{p}^{(2)} \in \mathcal{G}$  satisfying

$$\left( (\mathbf{p} - \mathbf{p}^{(1)})^T \cdot \mathbf{o}^{(\omega_{\mathcal{G}})} \right) \cdot \left( (\mathbf{p} - \mathbf{p}^{(2)})^T \cdot \mathbf{o}^{(\omega_{\mathcal{G}})} \right) \leq 0. \quad (4.2.7)$$

Condition (4.2.7) means that the parallel components of  $(\mathbf{p} - \mathbf{p}^{(1)})$  and of  $(\mathbf{p} - \mathbf{p}^{(2)})$  with respect to  $\mathbf{o}^{(\omega_{\mathcal{G}})}$  face opposite directions (or one of them has length zero). In other words, the set  $\mathbb{X}_{\mathcal{G}}$  just consists of pixels which are in the text range of  $\mathcal{G}$ . In Fig. 4.4c the set  $\mathbb{X}_{\mathcal{G}^{(2)}}$  of pixels for GT baseline 2 is shown (green points). It is easy to see, that the pixel of  $\mathcal{G}^{(4)}$  which are not green, are not in the text range of  $\mathcal{G}^{(2)}$ .

Let  $\mathbf{p} \in \mathbb{X}_{\mathcal{G}}$  be arbitrary.  $m_{\mathcal{G}}(\mathbf{p}) \in \mathcal{G}$  is the pixel for which the parallel component of  $(\mathbf{p} - m_{\mathcal{G}}(\mathbf{p}))$  with respect to  $\mathbf{o}^{(\omega_{\mathcal{G}})}$  has minimal length. Thus, it is defined as

$$m_{\mathcal{G}}(\mathbf{p}) := \arg \min_{\mathbf{q} \in \mathcal{G}} d_{in}(\mathbf{p}, \mathbf{q}; \mathbf{o}^{(\omega_{\mathcal{G}})}).$$

The minimum distance of  $\mathcal{G}$  to the "closest" polygonal chain is now defined as

$$d_{\mathcal{G}} := \min_{\mathbf{p} \in \mathbb{X}_{\mathcal{G}}} d_{off}(\mathbf{p}, m_{\mathcal{G}}(\mathbf{p}); \omega_{\mathcal{G}}). \quad (4.2.8)$$

Basically,  $d_{\mathcal{G}}$  is the minimal length of all the orthogonal components of  $(\mathbf{p} - m_{\mathcal{G}}(\mathbf{p}))$  with respect to  $\mathbf{o}^{(\omega_{\mathcal{G}})}$ , see Fig. 4.4c (green lines). For  $\mathbb{X}_{\mathcal{G}} = \emptyset$  there are no baselines which allow for a calculation of  $d_{\mathcal{G}}$ . In this case, its tolerance value is set to some default value (250 was chosen as a meaningful default regarding practical relevant scenarios).

The condition of Eq. (4.2.7) is mandatory to calculate reasonable distances. If  $\mathbb{X}_{\mathcal{G}}$  is not estimated carefully, the minimal distance calculation of Eq. (4.2.8) would suffer from pixels which are not in the text range of the regarded text line. E.g., the off-text distance of the yellow pixel in Fig. 4.4c to GT line 2 is significantly smaller than the reasonable minimum distance. Hence, it would falsify the statistics. Nevertheless, it is not in the range of  $\mathcal{P}$  (Eq. (4.2.7) is not fulfilled), therefore not in  $\mathbb{X}_{\mathcal{G}}$ , and consequently not regarded in Eq. (4.2.8).

The mean  $\bar{d}_{\mathbb{G}}$  is calculated for all  $d_{\mathcal{G}}$  ( $\mathcal{G} \in \mathbb{G}$ ) which have a value different to the default. Finally, the GT baseline dependent tolerance values are calculated as

$$t_{\mathcal{G}} := 0.25 \cdot \min(d_{\mathcal{G}}, \bar{d}_{\mathbb{G}}).$$

25% of the estimated interline distance yields a reasonable compromise between accuracy and flexibility. In Fig. 4.4d the red areas display the individual tolerance areas for the different GT baselines. Finally, the tolerance values along with the polygonal chain normalization allow for a certain margin of baselines. Thus, requirement (I) for the similarity score is met.

## Coverage Function

To evaluate (i) & (ii) (R-value & P-value) some kind of overlapping between GT and HYP has to be calculated. For this purpose we introduce the coverage function.

**Definition 4.2.6** (coverage function). The function  $\text{COV} : \mathfrak{P}_{BL} \times \mathfrak{P}_{BL} \times \mathbb{R} \rightarrow \mathbb{R}$  which is determined by Alg. 4 is called *coverage function*.

Basically, the coverage function  $\text{COV}(\mathcal{P}^{(1)}, \mathcal{P}^{(2)}, t)$  determines a value representing the fraction of a polygonal chain  $\mathcal{P}^{(1)}$  for which there is a pixel of another polygonal chain  $\mathcal{P}^{(2)}$  within a certain tolerance area determined by  $t$ . The coverage function is independent of the text orientation. As a result, requirement (II) is met, because the tolerance value calculation is also applicable for various text orientations.

Alg. 4 counts the number of pixels of  $\mathcal{P}^{(1)}$  for which there is a pixel of  $\mathcal{P}^{(2)}$  with a distance less than the given tolerance value  $t$ . Furthermore, a smooth (linear)

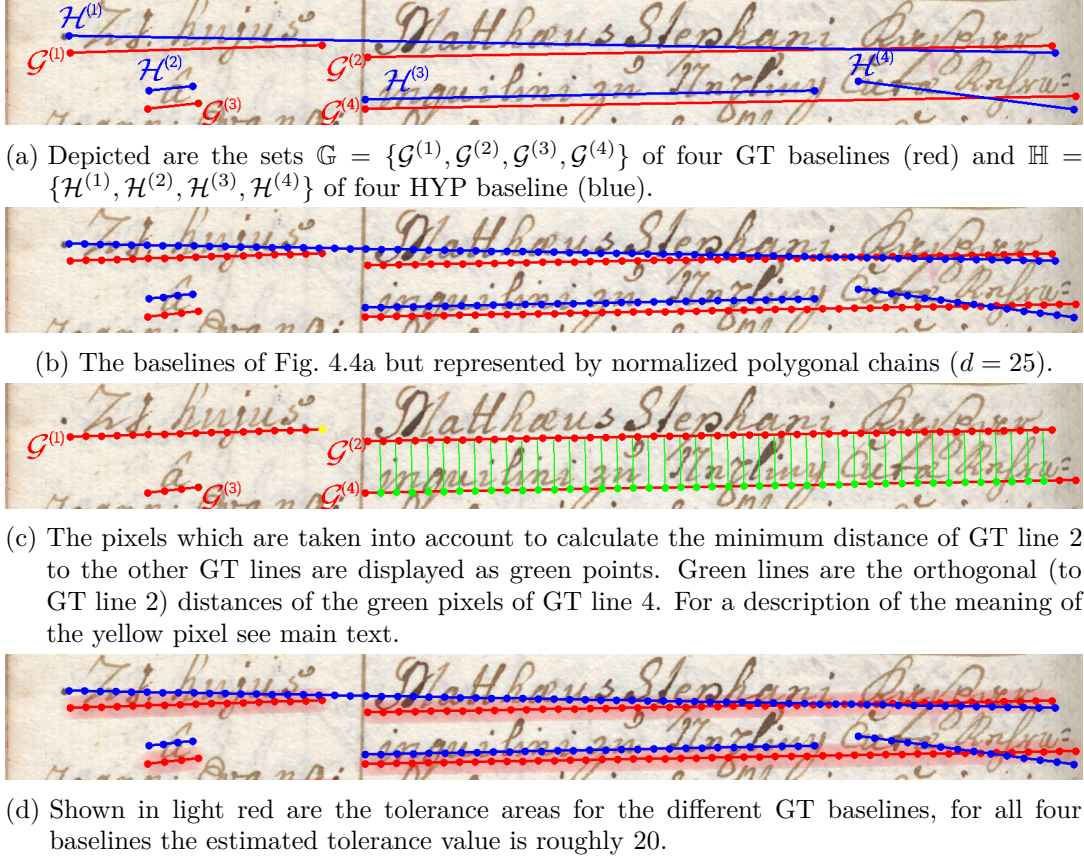


Figure 4.4.: **Baseline similarity score** – Depicted is a snippet of an example document image sampled from [Die+17b] along with GT and HYP baselines as well as intermediate steps of the evaluation scheme.

transition is performed for pixels of  $\mathcal{P}^{(1)}$  with a distance between  $t$  and  $3t$ . A pixel with a distance less than  $t$  counts 1, with a distance of  $1.5t$  it counts 0.75, with a distance of  $2t$  it counts 0.5, ... Finally, a pixel with a distance of  $3t$  and more counts 0. The resulting value is normalized by the number of pixels in  $\mathcal{P}^{(1)}$ .

Let  $\text{COV}_S : \mathfrak{P}_{BL} \times 2^{\mathfrak{P}_{BL}} \times \mathbb{R} \rightarrow \mathbb{R}$  be the generic extension of COV to a function accepting sets of polygonal chains as second argument. The minimum from line 3 in Alg. 4 is calculated over a set of polygonal chains instead of a single polygonal chain. To clarify the functionality of the coverage function, a few exemplary values are shown in Tab. 4.1. Note that the function COV is not commutative in the first two arguments.

### R-value, P-value, and F-value Calculation

Now, we will introduce the R-value and P-value, see (i) & (ii).

**Definition 4.2.7** (R-value). Let  $\mathbb{G}, \mathbb{H}$  be the sets of normalized GT and HYP

---

**Algorithm 4:** Coverage Function:  $c = \text{COV}(\mathcal{P}^{(1)}, \mathcal{P}^{(2)}, t)$ 


---

**input** : polygonal chains  $\mathcal{P}^{(1)}, \mathcal{P}^{(2)}$ , tolerance value  $t$   
**output**: coverage value  $c$

```

1  $c \leftarrow 0$ 
2 for  $p \in \mathcal{P}^{(1)}$  do
3    $d_{\min} \leftarrow \min_{q \in \mathcal{P}^{(2)}} \|p - q\|_2$ 
4   if  $d_{\min} \leq t$  then
5      $c \leftarrow c + 1$ 
6   else if  $d_{\min} \leq 3t$  then
7      $c \leftarrow c + \frac{3t - d_{\min}}{2t}$ 
8  $c \leftarrow \frac{c}{|\mathcal{P}^{(1)}|}$ 
return :  $c$ 

```

$\triangleright |\mathcal{P}^{(1)}|$  is the number of pixels in  $\mathcal{P}^{(1)}$

---

Table 4.1.: **Examples for the coverage function** – Example values of the coverage function for the normalized polygonal chains shown in Fig. 4.4b with a fixed tolerance value of 20 (as shown in Fig. 4.4d).  $\mathcal{G}^{(i)}$  means the normalized version of the  $i$ -th GT baseline.

$\mathcal{P}$	$\mathcal{Q}$	$\mathcal{R}$	$\text{COV}(\mathcal{P}, \mathcal{Q}, 20)$	$\text{COV}_S(\mathcal{P}, \{\mathcal{Q}, \mathcal{R}\}, 20)$
$\mathcal{H}^{(3)}$	$\mathcal{G}^{(4)}$	–	1.0	–
$\mathcal{G}^{(4)}$	$\mathcal{H}^{(3)}$	$\mathcal{H}^{(4)}$	0.65	0.96
$\mathcal{G}^{(3)}$	$\mathcal{H}^{(2)}$	–	0.76	–
$\mathcal{H}^{(1)}$	$\mathcal{G}^{(1)}$	$\mathcal{G}^{(2)}$	0.26	0.95

baselines. Let  $t_{\mathcal{G}}$  denote the tolerance value for  $\mathcal{G} \in \mathbb{G}$ . The value defined as

$$R(\mathbb{G}, \mathbb{H}) := \frac{\sum_{\mathcal{G} \in \mathbb{G}} \text{COV}_S(\mathcal{G}, \mathbb{H}, t_{\mathcal{G}})}{|\mathbb{G}|}. \quad (4.2.9)$$

is called *R-value*.

The R-value indicates for what fraction of the GT baselines there are detected HYP baselines within a certain tolerance area. Because no alignment between GT and HYP baselines is enforced, segmentation (page layout) errors are not penalized at all. The segmentation errors are penalized in the P-value. To penalize segmentation errors the GT and HYP baselines have to be aligned.

**Definition 4.2.8** (alignment). A subset of the Cartesian product  $\mathbb{M} \subset \mathbb{G} \times \mathbb{H}$  is



called *alignment* between  $\mathbb{G}$  and  $\mathbb{H}$  iff

$$\begin{aligned} |\{(\tilde{\mathcal{G}}, \mathcal{H}) \in \mathbb{M} \mid \tilde{\mathcal{G}} = \mathcal{G}\}| &\leq 1 \quad \forall \mathcal{G} \in \mathbb{G} \\ |\{(\mathcal{G}, \tilde{\mathcal{H}}) \in \mathbb{M} \mid \tilde{\mathcal{H}} = \mathcal{H}\}| &\leq 1 \quad \forall \mathcal{H} \in \mathbb{H} \end{aligned}$$

hold.

Hence, a subset of the Cartesian product is called alignment if each element of  $\mathbb{G}$  as well as of  $\mathbb{H}$  occurs at most once in the pairs.

**Definition 4.2.9** (P-value). Let  $\mathbb{G}, \mathbb{H}$  be the sets of normalized GT and HYP baselines. Let  $t_{\mathcal{G}}$  denote the tolerance value for  $\mathcal{G} \in \mathbb{G}$ . The value defined as

$$P(\mathbb{G}, \mathbb{H}) := \max_{\substack{\mathbb{M} \subset \mathbb{G} \times \mathbb{H} \\ \text{is alignment}}} \frac{\sum_{(\mathcal{G}, \mathcal{H}) \in \mathbb{M}} \text{COV}(\mathcal{H}, \mathcal{G}, t_{\mathcal{G}})}{|\mathbb{H}|}. \quad (4.2.10)$$

is called *P-value*.

An alignment ensures that segmentation errors are penalized. E.g., if a text line is split into two equally sized parts an R-value of 1.0 is calculated (the two detected polygonal chains cover the entire GT baseline). But the expected P-value is 0.5 (the GT baseline is aligned with exactly one of the HYP polygonal chains with a coverage of 1, this is divided by 2, because there are two HYP polygonal chains).

We propose a greedy P-alignment to determine the maximum in Eq. (4.2.10). This is due to the fact that there is a huge number of possible alignments.

**Lemma 4.2.10.** Let  $|\mathbb{G}| = M$  and  $|\mathbb{H}| = K$  be the number of GT and HYP baselines. There are

$$N = \sum_{n=0}^{\min(M,K)} n! \cdot \binom{M}{n} \cdot \binom{K}{n} \geq 2^{\min(M,K)}$$

*different alignments.*

*Proof.* Let  $N$  denote the total number of different alignments. We call an alignment  $\mathbb{M}$  an  $n$ -alignment iff  $|\mathbb{M}| = n$  holds. Let  $N_n$  denote the total number of  $n$ -alignments. It is easy to see (Def. 4.2.8), that  $N_n > 0$  only if  $n \in \{0, \dots, \min(M, K)\}$ . Consequently,

$$N = \sum_{n=0}^{\min(M,K)} N_n \quad (4.2.11)$$

holds. Now, we will count all possible  $n$ -alignments. Therefore, we will set up an arbitrary  $n$ -alignment. Because  $M \cdot K = |\mathbb{G} \times \mathbb{H}|$  holds, there are  $M \cdot K$  possible

first elements for an alignment. This has to be multiplied by the number of possible second elements, ... Because of the uniqueness of each element in an alignment there are  $(M - 1) \cdot (K - 1)$  possible second elements and so on. Since the ordering is not of importance (an alignment is a set), the resulting number has to be divided by  $n!$ . With the falling factorial  $n^{\underline{m}} := \frac{n!}{(n-m)!}$  we get

$$\begin{aligned} N_n &= \frac{1}{n!} \prod_{i=0}^{n-1} (M - i) \cdot (K - i) \\ &= \frac{1}{n!} \cdot M^{\underline{n}} \cdot K^{\underline{n}} \\ &= n! \cdot \binom{M}{n} \cdot \binom{K}{n} \end{aligned} \tag{4.2.12}$$

The stated equality directly follows from (4.2.11) & (4.2.12). The inequality is a consequence of

$$\sum_{k=0}^n \binom{n}{k} = 2^n$$

and equality holds for  $M = K = 0$ . □

Even for reasonable numbers  $M = 50$  and  $K = 50$  the number of possible alignments exceeds (by far)  $2^{50}$ . Hence, a direct optimization by testing all possible alignments is impossible or at least cumbersome. Thus, the greedy P-alignment is a good compromise and it yields suitable alignments in all our practical cases.

**Definition 4.2.11** (greedy P-alignment). The set of pairs  $\mathbb{M} \subset \mathbb{G} \times \mathbb{H}$  defined as

$$\mathbb{M} := \text{ALIGN}(\mathbb{G}, \mathbb{H}) \quad (\text{see Alg. 5})$$

is called *greedy P-alignment*.

We have described the computations rules for the R-value and for the P-value (i) & (ii). These rules do not rely on any reading order nor on a binarization of the underlying document image (nor on any pixel information of the image). Consequently, the requirement (III) is met. That means all requirements as stated in the introduction are satisfied. Finally, the R-value and the P-value are combined by means of their harmonic mean to constitute the F-value. The F-value is employed as (baseline) similarity score throughout this work.

**Definition 4.2.12** (F-value). Let  $\mathbb{G}, \mathbb{H}$  be the sets of normalized GT and HYP baselines. The harmonic mean of R-value and P-value defined as

$$F(\mathbb{G}, \mathbb{H}) := \frac{2 \cdot R(\mathbb{G}, \mathbb{H}) \cdot P(\mathbb{G}, \mathbb{H})}{R(\mathbb{G}, \mathbb{H}) + P(\mathbb{G}, \mathbb{H})}$$

is called *F-value*.

**Algorithm 5:** Greedy P-Alignment Function:  $\mathbb{M} = \text{ALIGN}(\mathbb{G}, \mathbb{H})$ 


---

**input** : GT and HYP sets  $\mathbb{G} = \{\mathcal{G}^{(1)}, \dots, \mathcal{G}^{(M)}\}$ ,  $\mathbb{H} = \{\mathcal{H}^{(1)}, \dots, \mathcal{H}^{(K)}\}$   
**output**: greedy P-alignment  $\mathbb{M}$

```

1  $\mathbb{M} \leftarrow \emptyset$ 
2  $C \leftarrow$  coverage matrix defined by  $C_{i,j} = \text{COV}(\mathcal{H}^{(i)}, \mathcal{G}^{(j)}, t_{\mathcal{G}^{(j)}}) \forall i \in [K], j \in [M]$ 
3 while  ${}_1C > 0 \wedge {}_2C > 0$  do
4    $(i, j) \leftarrow \arg \max_{i \in [K], j \in [M]} C_{i,j}$ 
5   if  $m > 0$  then  $\triangleright$  create an aligned pair
6      $\mathbb{M} \leftarrow \mathbb{M} \cup (\mathcal{G}^{(j)}, \mathcal{H}^{(i)})$ 
7      $C \leftarrow$  take  $C$  and delete row  $i$  and col  $j$ 
8   else
      $\perp$  return :  $\mathbb{M}$ 
return :  $\mathbb{M}$ 

```

---

**Examples**

Results for different subsets of the GT and HYP baselines of Fig. 4.4a are shown in Tab. 4.2 and explained in the following.

Table 4.2.: **Examples for the R-, P-, and F-values** – The values for different subsets of the GT and HYP baselines shown in Fig. 4.4a are calculated. A fixed tolerance value of 20 is used for the calculations.

Ex.	$\mathbb{G}$	$\mathbb{H}$	R	P	F
1	$\{\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \mathcal{G}^{(3)}, \mathcal{G}^{(4)}\}$	$\{\mathcal{H}^{(1)}, \mathcal{H}^{(2)}, \mathcal{H}^{(3)}, \mathcal{H}^{(4)}\}$	0.91	0.61	0.73
2	$\{\mathcal{G}^{(2)}, \mathcal{G}^{(3)}, \mathcal{G}^{(4)}\}$	$\{\mathcal{H}^{(1)}, \mathcal{H}^{(2)}, \mathcal{H}^{(3)}, \mathcal{H}^{(4)}\}$	0.9	0.61	0.73
3	$\{\mathcal{G}^{(1)}, \mathcal{G}^{(3)}, \mathcal{G}^{(4)}\}$	$\{\mathcal{H}^{(1)}, \mathcal{H}^{(2)}, \mathcal{H}^{(3)}, \mathcal{H}^{(4)}\}$	0.89	0.51	0.65
4	$\{\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \mathcal{G}^{(3)}, \mathcal{G}^{(4)}\}$	$\{\mathcal{H}^{(2)}, \mathcal{H}^{(3)}\}$	0.35	0.88	0.5
5	$\{\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \mathcal{G}^{(3)}, \mathcal{G}^{(4)}\}$	$\{\mathcal{H}^{(2)}, \mathcal{H}^{(3)}, \mathcal{H}^{(4)}\}$	0.43	0.6	0.5

The small difference between Ex. 1 and Ex. 2 is due to the fact, that in both cases  $\mathcal{H}^{(1)}$  is aligned with  $\mathcal{G}^{(2)}$  for the P-value calculation. Hence, there is no effect on the P-value if  $\mathcal{G}^{(1)}$  is removed. The R-value is nearly the same, because  $\mathcal{G}^{(1)}$  and  $\mathcal{G}^{(2)}$  are both completely covered by  $\mathcal{H}^{(1)}$ . By removing  $\mathcal{G}^{(2)}$  instead of  $\mathcal{G}^{(1)}$  (Ex. 3),  $\mathcal{H}^{(1)}$  is now aligned with  $\mathcal{G}^{(1)}$  yielding a lower P-value, because  $\mathcal{G}^{(2)}$  covers much more of  $\mathcal{H}^{(1)}$  than  $\mathcal{G}^{(1)}$ . In Ex. 4 one gets a high P-value, because the remaining HYP baselines are very well covered by the GT baselines. Of course, the R-value is increased by adding  $\mathcal{H}^{(4)}$  (Ex. 5), but the P-value is decreased. This is due to the fact that  $\mathcal{H}^{(3)}$  is aligned with  $\mathcal{G}^{(4)}$  (as in Ex. 4) and  $\mathcal{H}^{(4)}$  is not aligned at all and gets a P-value of 0.

**Remark 4.2.13.** Short text lines have the same impact as long ones for the R-value as well as the P-value (and consequently the F-value). This is because in Eq. (4.2.9) and Eq. (4.2.10) the line specific R- and P-values are divided by the number of GT and HYP lines, respectively. This prevents an underestimation of the importance of short text lines, which often contain essential information in the context of historical documents, e.g. ages, dates, sex.

### Multi Page Evaluation

Since some datasets, e.g. the cBad dataset [Die+17b], are very heterogeneous, it is meaningful to evaluate each page on its own and to average these page-wise results. This prevents an overbalance of pages with dozens of baselines (like pages containing a table) and yields results representing the robustness of the evaluated algorithms over various scenarios. This is already taken into account in the definition of the test score, see Def. 1.3.10. Hence, it is also valid for the (baseline) test score.

### Computational Effort

In this section, we investigate the complexity of the proposed similarity score. Because the number of GT and HYP baselines as well as their shape are given by an operator and the method to be evaluated, the complexity is investigated for the distance parameter  $d$  in the polygonal chain normalization. This is a free parameter of the evaluation scheme.

**Corollary 4.2.14.** *Let  $\mathbb{G}, \mathbb{H}$  be the sets of normalized GT and HYP baselines. Let  $n = 1/d$  denote the number of pixels per unit of length. The (baseline) similarity score  $(F(\mathbb{G}, \mathbb{H}))$  is in  $\mathcal{O}(n^2)$ .*

*Proof.* The complexity reduces to the complexity of the coverage function, see Alg. 4. The summations and loops in Eq. (4.2.9), Eq. (4.2.10), and in Alg. 5 just depend on  $|\mathbb{G}|, |\mathbb{H}|$  and therefore contribute to the constant term. Line 2 of Alg. 4 is linear in  $n$  and line 3 is also linear in  $n$  (for the proposed straight-forward implementation). Hence, the coverage function is in  $\mathcal{O}(n^2)$ .  $\square$

Consequently, the distance parameter has (in theory) a quadratic influence on the computation time of the similarity score. Of course, a coarse resolution (high value  $d$ ) results in an inaccurate similarity score, see Fig. 4.5.

In Tab. 4.3 the F-value and the corresponding computation time are shown for different distance parameters. Finally, a distance value of  $d = 5$  yields a good compromise between accuracy and computational time and is used in this work.

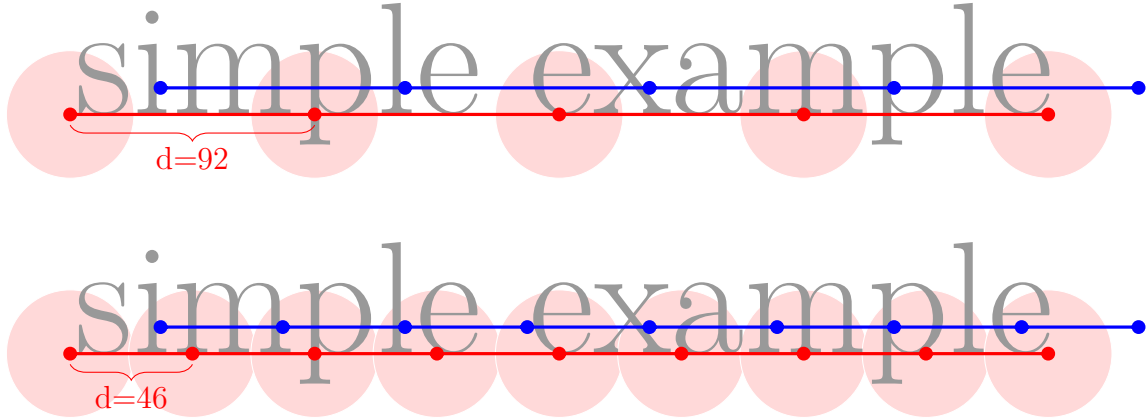


Figure 4.5.: **Effect of the normalization on the similarity score** – Shown are two different scenarios for the similarity score calculation for different distance parameters in the polygonal chain normalization. GT baselines are red, HYP baselines blue. The red circles (radius of  $3t$  with a tolerance value  $t$ , cf. Alg. 4) are the areas of interest for the coverage function. For  $d = 92$  the similarity score would be 0, because the HYP pixels are all outside of the areas of interest. On the contrary, the result for  $d = 46$  would be reasonable.

Table 4.3.: **Relation between F-value and computational time** – Shown are F-values and the time required for their computation for several distance parameters. The experiments were performed for an exemplary hypothesis set for the cBad test set (complex track), see [Die+17a]. The test set is composed of 1010 pages of heterogeneous historical documents.

$d$	1	3	5	10	25	50	100
F-value	0.9259	0.9244	0.9231	0.9186	0.8935	0.803	0.6999
time in s	460	94.9	40.9	17.4	9.0	6.4	5.2

### 4.3. Baseline Detection Method

After motivating and introducing the problem of baseline detection as well as the (baseline) similarity score to evaluate the quality of baseline detection methods, we will describe the developed baseline detection method. The following sections orient towards [Grü+18b], some paragraphs and figures are citations of the mentioned work. Nevertheless, major parts were substantially rewritten to incorporate the cited work into this thesis.

As already mentioned in Chap. 1 & 2, deep learning based methods became omnipresent in the document analysis community within the last years. Such techniques were recently used to solve several different problems such as binarization [Vo+17],

page boundary extraction [Ten+17], page segmentation [Che+17] or text line detection [Ren+17]. Nevertheless, usually either classical image processing based approaches or deep learning based approaches are proposed in the literature. The presented work to our knowledge is the first which uses a two-stage method, combining deep learning strategies and state-of-the-art image processing based techniques.

For this purpose, we propose a newly designed fully convolutional network (FCN), see Sec. 3.1. The proposed FCN, the so-called ARU-Net, is an extension of the U-Net [RFB15]. The fully convolutional U-Net is extended by incorporating residual blocks [He+16] to increase its representative power. Furthermore, a spatial attention mechanism is developed which allows the ARU-Net to focus on image content at different positions and scales. The network is designed to process the entire, arbitrarily-sized image at once to take account of all spatial context. The ARU-Net is universal in a way that it could be used to tackle any pixel labeling task.

In this work, it is trained in a fully supervised fashion to classify each pixel to belong to one of the following classes: baseline, separator or other. The separator class is introduced to explicitly predict beginning and end of each text line and not just rely on the information implicitly given by the baseline class. This is advantageous for text lines which are close together but have to be separated, e.g., those belonging to different columns.

The network output serves as input for an image processing based bottom-up clustering approach. This approach utilizes so-called states of superpixels [RKC14], which encode local text orientation and interline distances. This second stage allows for an error correction of the network output by incorporating domain knowledge based on assumptions, which hold for text lines in general. Additionally, it is easily possible to incorporate the separator information, which allows for a handling of documents with complex layouts, e.g., images containing tables or marginalia.

We show that the presented approach outperforms state-of-the-art methods on three different datasets. A relative F-value (cf. Sec 4.2) error (the gap to 1.0) reduction of at least 24% is achieved for the cBAD dataset [Die+17b]. This dataset is composed of images of nine different archives and libraries from all over Europe and is therefore – in the opinion of the authors – the most representative and heterogeneous freely available dataset. Especially, for the complex track, which contains mostly documents with complex layouts, the average F-value is increased from 0.859 to 0.922.

In the following, we introduce the two-stage method for baseline detection, see Fig. 4.6. The first stage relies on an FCN – the ARU-Net – and performs a pixel labeling. The pixel labeling can be seen as some kind of goal-oriented binarization. Instead of detecting all foreground elements, it restricts itself to those elements which are of interest for the specific task. The second stage performs a superpixel extraction on the first stage’s output. These superpixels are further clustered to build baselines.

First, a detailed description of the proposed ARU-Net is given. Finally, the superpixel extraction and clustering approach is described.

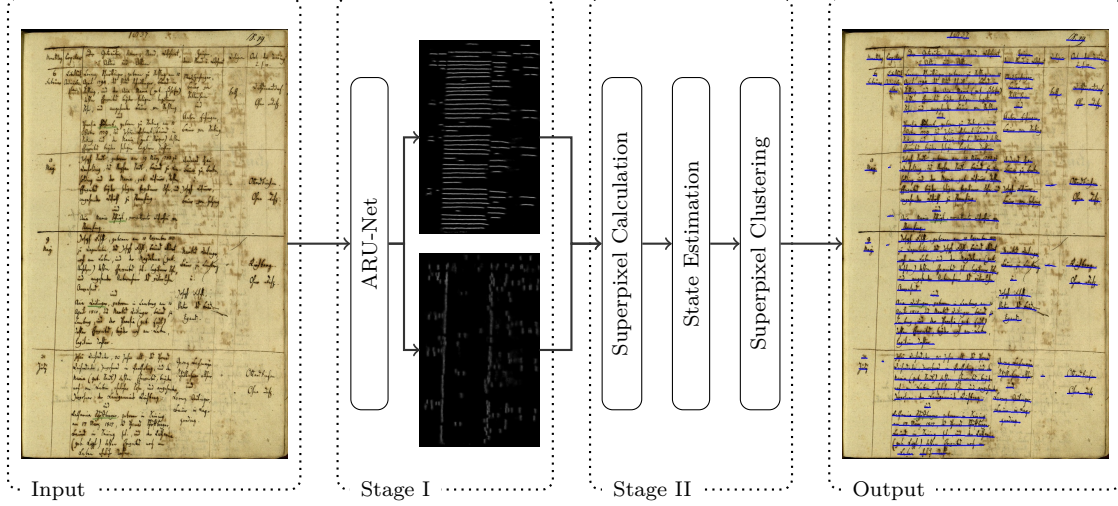


Figure 4.6.: **Two-stage workflow to detect baselines** – The first stage utilizes a deep artificial neural network to perform a pixel labeling. The result of Stage I is the input for an image processing based method in Stage II. This method clusters superpixels to build baselines. The image is sampled from the cBad complex test set [Grü+18a].

#### 4.3.1. Stage I: ARU-Net

Typically, layout analysis algorithms directly work on the input image  $I$  or on a binarized version of it [AS14]; [RKC14]; [NG09]; [SAE14]; [Gar+12]; [Grü+17]. Instead, we employ a more goal-oriented transformation of the input image utilizing an FCN, which is trained in a supervised manner to assign a certain class to each pixel like in [RFB15]; [LSD15]; [NHH15]. This is often referred to as pixel labeling or semantic segmentation. We will introduce the problem of pixel labeling utilizing FCNs, followed by a description of the proposed ARU-Net architecture.

#### Pixel Labeling – Problem Formulation

Essentially, the pixel labeling problem is a spatial classification problem as introduced in Def. 3.1.28. Here, we introduce the neural pixel labeler to solve the problem and the pixel ground truth to train the corresponding ANN. In Sec. 3.1 the FCN was introduced. Basically, it is a parametrized function  $\Phi(\cdot; \theta) : \mathbb{R}^{* \times * \times z_{in}} \rightarrow \mathbb{R}^{* \times * \times z_{out}}$ . This function maps from a 3-dim array with arbitrary spatial dimensions and fixed representative depth to a 3-dim array of dimensions determined by the input's dimension and of the hyperparameters of the FCN. It is easy to see, that any image

can be represented as a 3-dim array with a representative depth of 1. Hence, one can state  $\mathcal{I} \subset \mathbb{R}^{* \times * \times 1}$ . Furthermore,  $\mathcal{I}^{|\mathcal{C}|} \subset \mathbb{R}^{* \times * \times |\mathcal{C}|}$  holds. Therefore, images are feasible inputs to FCNs and the introduced neural pixel labeler is just a special FCN.

**Definition 4.3.1** (neural pixel labeler). Let  $\mathcal{C} = \{c_1, \dots, c_n\}$  be a set of distinct classes. An FCN  $\Phi(\cdot; \boldsymbol{\theta}) : \mathcal{I} \rightarrow \mathcal{I}^{|\mathcal{C}|}$  is called *neural pixel labeler (NPL)* iff

$$\Phi(I; \boldsymbol{\theta}) = \mathbf{C} \in [0, 1]^{I \times 2I \times |\mathcal{C}|} \text{ subject to } \sum_{k=1}^{|\mathcal{C}|} \mathbf{C}_{i,j,k} = 1 \quad \forall i \in [1I], j \in [2I]$$

holds.

The NPL is parametrized by the model parameters  $\boldsymbol{\theta} \in \mathbb{R}^N$  and it produces a prediction over all pixels and all possible classes for  $I \in \mathcal{I}$ . Hence,  $\mathbf{C}_{:,j,k} \in \mathcal{I}$  denotes the image which encodes the pixel-wise prediction (probability) for the  $k$ -th class. Hence, the NPL aims at solving the spatial classification task, see Def. 3.1.28.

**Definition 4.3.2** (pixel ground truth). A 3-dim array  $\mathbf{G}^{(I)} \in \mathcal{I}^{|\mathcal{C}|}$  is called *pixel ground truth* (for image  $I$ ) if it assigns exactly one class (one-hot-encoding) to each pixel such that

$$\forall i \in [1I], j \in [2I] \quad \exists! k \in \{1, \dots, |\mathcal{C}|\} : \quad \mathbf{G}_{i,j,k}^{(I)} = 1 \wedge \mathbf{G}_{i,j,\tilde{k}}^{(I)} = 0 \quad (\forall \tilde{k} \neq k).$$

Following the methodology of Sec. 3.1.3, we aim for an NPL which was tuned on a training set and performs optimal on a test set. Because of the one-hot-encoding of the pixel ground truth the spatial negative log-likelihood ( $L_{SNLL}$ ) of Def. 3.1.30 can be utilized to evaluate the NPL and to train its model parameters, see Sec. 3.1.4.

## ARU-Net – Architecture

The ARU-Net is a special NPL and is described in this section. Within the last few years, different architectures were proposed for the pixel labeling task. Most of them are based on FCNs. A direct application of FCNs for pixel labeling is presented in [LSD15]. The presented FCN combines local features to produce more meaningful high level features using pooling layers. These pooling layers reduce the spatial dimension of the input, cf. Def. 3.1.15. Thus, the result suffers from a coarse resolution. Noh et al. [NHH15] tackle this problem by applying a deconvolutional network on the subsampled output of the FCN. The deconvolutional layer increases the resolution, such that the output has the same spatial dimension as the input. The U-Net proposed in [RFB15] furthermore introduces shortcuts between layers of the same spatial dimension. This allows for an easier combination of local low level features and global higher-level features. Additionally, error propagation for deep structures is facilitated and the vanishing gradient problem [GB10] is reduced.



The U-Net is the basis for the proposed ARU-Net. We extend the U-Net by two more key concepts – spatial attention (A) and depth (residual structures (R)) to be described below. Remarkably, in contrast to the U-Net proposed in [RFB15], we perform border padding, see Def. 3.1.10. Hence, the spatial dimensions in each scale space of the U-Net are all the same, see Fig. 4.7 for a schematic representation of a U-Net. Thus, the output of the U-Net comprises feature maps ( $z$  feature maps in Fig. 4.7) of the same spatial dimension as the input. Hence, the U-Net becomes an NPL as defined in Def. 4.3.1 by adding a convolutional layer with convolutional softmax activation (to get pixel-wise predictions) which processes the U-Net output and distinguishes between the different classes in  $\mathfrak{C}$ .

**Remark 4.3.3.** If the presented architectures are used for pixel labeling, it is implicitly assumed that such a convolutional softmax layer is always added to generate per class probabilities at pixel level.

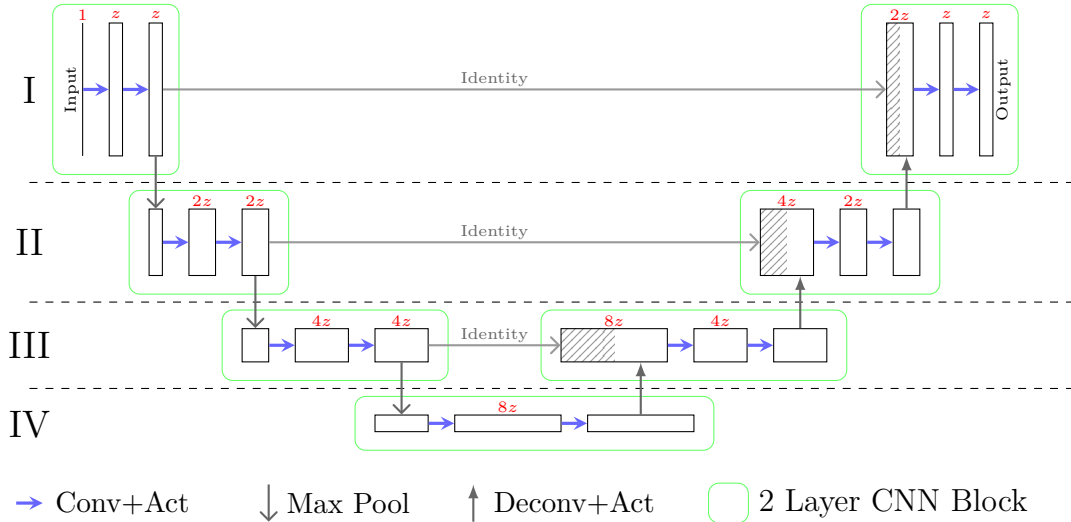


Figure 4.7.: **U-Net** – The input is an image of arbitrary spatial dimension. "Act" is the activation function thus the rectangles represent sets of activation maps. Each rectangle represents a 3-dim array ( $\in \mathbb{R}^{* \times * \times *}$ ) of certain dimensionality. Within each scale space (roman numbers) the feature map widths and heights are constant (encoded by the height of the rectangles). The number of feature maps  $z$  is pictured by the width of the rectangles. Between adjacent scale spaces the spatial dimension decreases by a certain factor (2 in the figure) and the representative depth (number of feature maps) increases by the same factor.

He et al. [He+16] introduce very deep neural networks which are still trainable and yield state-of-the-art results. This is achieved using so-called *residual blocks*. The basic idea behind residual blocks is: If a set of stacked layers can approximate the function  $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$   $f(\mathbf{x})$ , it can also approximate the residual function  $h(\mathbf{x}) = f(\mathbf{x}) - \mathbf{x}$ . Consequently, one gets an approximation of  $f$  by  $f(\mathbf{x}) \approx \mathbf{x} + h(\mathbf{x})$ .

There are various different forms of residual blocks. The one used within this work is depicted in Fig. 4.8. As a result, residual blocks introduce shortcuts. These shortcuts enable the error backpropagation as well as an identity propagation even for very deep structures. Hence, the vanishing gradient problems are reduced [He+16].

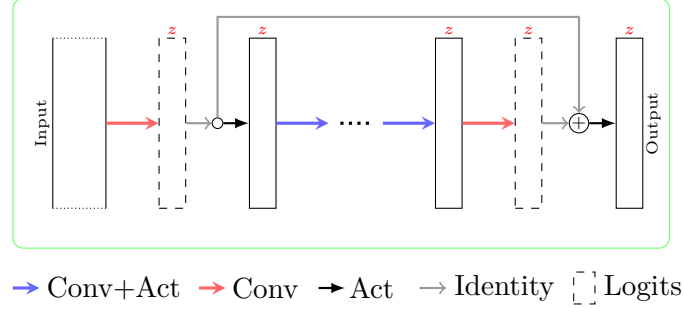


Figure 4.8.: **Residual Block** – The input is convolved and the resulting 3-dim array (the maps before passed through an activation function are referred to as logits) is used twice. At the first branch it is passed through the activation function and further processed by several convolution layers. At the second branch it is directly fed into a summation node. After a point-wise summation of the two logit maps an activation function is applied. The shortcut enables for an easy identity propagation and error backpropagation. Arbitrarily many inner layers are possible.

**Definition 4.3.4** (RU-Net). An *RU-Net* is a U-Net with residual blocks.

That means, each of the 2-layer CNN blocks in Fig. 4.7 is replaced by a residual block as in Fig. 4.8.

Now, we introduce a pixel-wise (spatial) attention mechanism to explicitly incorporate the capability to handle various font sizes (especially mixed font sizes on a single page). Attention mechanisms gained a lot of interest in the last years. In the survey of Wang et al. [WT16], they distinguish between hard-attention mechanisms and soft-attention mechanisms. Hard-attention mechanisms, e.g., [M+14]; [J+15]; [CCB15], pay attention to a certain area of the input. They are usually cumbersome to train. For instance, the network architecture proposed in [M+14] comprises some stochastic component. Hence, it is not trainable using standard backpropagation techniques, instead reinforcement learning has to be utilized. In contrast, soft-attention based mechanisms make use of a weighted attention over the entire input [Blu16]; [BLM17]; [BCB14]; [Cha+16]. I.e., each input element is multiplied with some scalar  $\in [0, 1]$ , a multiplication with 1 means full attention is paid to this input element, a multiplication with 0 means no attention is paid to this input element. The attention is usually encoded via an attention vector/map which is calculated given the input or some other features. Consequently, no hard decisions have to be made and the architectures are trainable with standard back

propagation. Xu et al. [Xu+15] demonstrates that the combination of both types of attention mechanisms could also yield satisfying results.

In this work, we will utilize a soft-attention based approach. For this purpose, we introduce an attention network (*A-Net*). The A-Net is a multi-layer CNN which generates a **single** output feature map – the attention map. The A-Net will be applied along with the RU-Net on the input image at different scales, the same network weights are used on all scales (weight sharing). Specially, a scale pyramid is built by downscaling the input image  $I = I^{(1)}$  several times. The resulting (scaled) images  $I^{(1)}, I^{(2)}, I^{(4)}, I^{(8)}, \dots, I^{(s)}$  (superscripts denote the scaling factors) are fed into the RU-Net and the A-Net. Trainable deconvolutional layers (of corresponding scales) are applied on the outputs of the RU- and the A-Net to obtain feature maps of spatial dimensions equal to the inputs.

$A^{(1)}, \dots, A^{(s)}$  denote the up-sampled feature map of the A-Net because the representative depth of the A-Net output is 1 they are denoted in the matrix notation (not bold).  $\mathbf{RU}^{(1)}, \dots, \mathbf{RU}^{(s)}$  are the features maps of the RU-Net, respectively. Let the representative depth of the output of the RU-Net be  $z$ . After applying a pixel-wise softmax normalization for the attention maps

$$\hat{A}_{i,j}^{(k)} = \frac{\exp(A_{i,j}^{(k)})}{\sum_{l \in \{1,2,4,\dots,s\}} \exp(A_{i,j}^{(l)})} \quad \forall k \in \{1,2,4,\dots,s\}, i \in [{}_1A^{(k)}], j \in [{}_2A^{(k)}]$$

the normalized attention maps  $\hat{A}^{(k)}$  sum to one (pixel-wise). The feature maps  $\mathbf{RU}^{(k)}$  are combined like in

$$\mathbf{ARU}_{::,l} = \sum_{k \in \{1,2,4,\dots,s\}} \mathbf{RU}_{::,l}^{(k)} \odot \hat{A}^{(k)} \quad \forall l \in [z].$$

where  $\odot$  is the Hadamard product (element-wise multiplication).  $\mathbf{ARU}$  is the input for the softmax activated convolutional layer. Hence, the ARU-Net becomes an NPL, see Rem. 4.3.3.

**Definition 4.3.5** (ARU-Net). An RU-Net incorporating the described spatial attention mechanism is called *ARU-Net*, see Fig. 4.9.

The point-wise multiplication combined with the pixel-wise attention maps allow the ARU-Net to pay attention in different scales at different positions of the image. In Fig. 4.9 one can see that this behavior was indeed learned by the network. It seems like the RU-Net is specialized on a certain font size and the A-Net distinguishes between areas of different font sizes (bright and dark areas).

The ARU-Net as introduced can be used for any pixel labeling task, e.g., binarization, page detection and page segmentation. The purpose of the ARU-Net is defined and fixed by the number of classes and the ground truth data provided for training. In this work, we limit ourselves to the baseline detection problem introduced

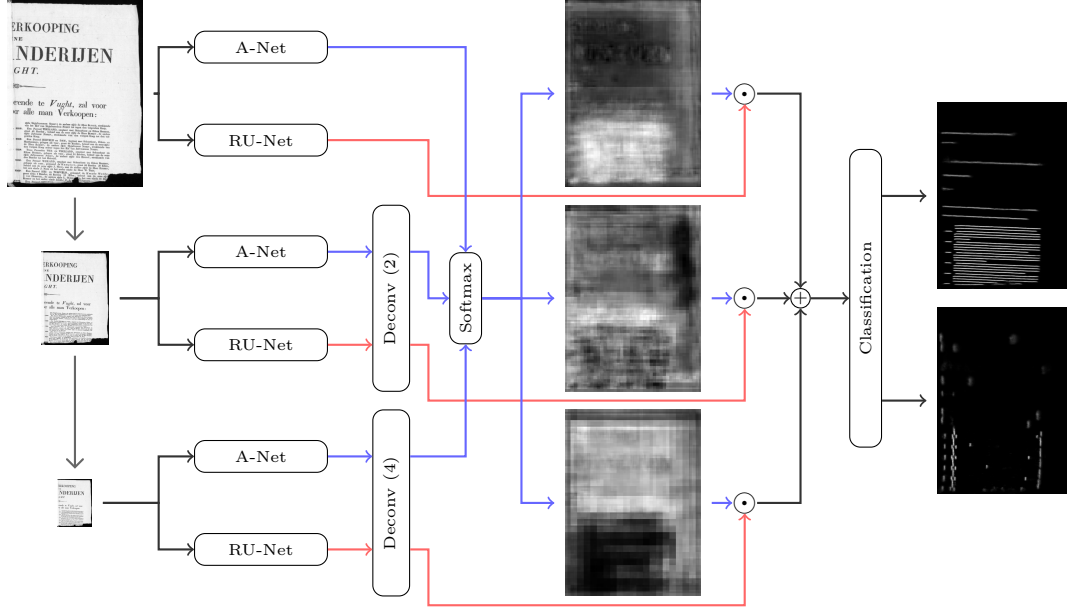


Figure 4.9.: **ARU-Net** – The input image and its downscaled versions are fed into the A-Net and R-U-Net (weight sharing accross different scales). The results for the lower resolutions are deconvolved. The attention maps are passed through a softmax normalization. The brighter the map at a certain position the more attention is paid to that position at the corresponding scale. The attention maps are point-wise muliplied with the feature maps of the RU-Net. The results are summed and a classification is performed.

in Sec. 4.1. For this purpose, we introduce three different classes: baseline (bl), separator (sep) and other ( $\emptyset$ ). The separators mark beginning and end of each text line. Although, the separator information is implicitly encoded by the baselines, it is advantageous to explicitly introduce it as possible classification result. Especially, for baselines which are close together, e.g., such belonging to two adjacent columns, this approach helps to avoid segmentation errors, see Sec. 4.4.

The  $L_{SNLL}$  loss functions requires per-pixel one-hot encoded GT information. Hence, pixel GT of Def. 4.3.2 is required to train the ARU-Net. In the baseline detection problem (see Sec. 4.1) the baseline GT is encoded by polygonal chains. Thus, the pixel GT for the classes  $\mathfrak{C} = \{\text{bl}, \text{sep}, \emptyset\}$  has to be produced. We propose an algorithm, see Alg. 6, to automatically generate the pixel GT for all three classes given solely the baseline GT. Hence, no additional human effort is necessary to generate the required pixel GT.

A sample image with baseline GT along with its generated pixel ground truth is depicted in Fig. 4.10. The prediction of a trained ARU-Net for this sample image is shown in Fig. 4.11a.

---

**Algorithm 6:** Pixel Ground Truth Generation:  $\mathbf{G} = PGTG(I, \tilde{I}_I(I))$ 


---

**input** : image  $I$ , baseline ground truth  $\mathbb{G} = \tilde{I}_I(I)$   
**output**: pixel ground truth  $\mathbf{G}$

```

1  $B, S, N \leftarrow 0$   $\triangleright$  of dimension  ${}_1I \times {}_2I$ 
2 for  $\mathcal{G} = (\mathbf{p}^{(1)}, \dots, \mathbf{p}^{(n)}) \in \mathbb{G}$  do
3    $\omega_{\mathcal{G}} \leftarrow$  polygonal chain orientation of  $\mathcal{G}$   $\triangleright$  see Def. 4.2.2
4    $d_{\mathcal{G}} \leftarrow$  interline distance of  $\mathcal{G}$   $\triangleright$  see Def. 4.3.9
5    $\mathcal{P}^{(b)} \leftarrow$  polygonal chain of length  $d_{\mathcal{G}}$  and orient.  $\omega_{\mathcal{G}} + 90^\circ$  centered at  $\mathbf{p}^{(1)}$ 
6    $\mathcal{P}^{(e)} \leftarrow$  polygonal chain of length  $d_{\mathcal{G}}$  and orient.  $\omega_{\mathcal{G}} + 90^\circ$  centered at  $\mathbf{p}^{(n)}$ 
7   draw  $\mathcal{P}^{(b)}$  and  $\mathcal{P}^{(e)}$  in  $S$   $\triangleright$  draw: set all pixels of the normalized
   polygonal chain to 1.0 in the image
8   draw  $\mathcal{G}$  in  $B$ 
9  $E \leftarrow 3 \times 3$  matrix of ones
10  $S \leftarrow S \oplus E$ 
11  $B \leftarrow (B \oplus E) \wedge \neg S$ 
12  $N \leftarrow \neg S \wedge \neg B$ 
return :  $\mathbf{G} \leftarrow (B, S, N)$ 

```

---

### 4.3.2. Stage II: Baseline Estimation

This subsection describes the second stage of the proposed baseline detection method. Baselines are estimated given the output of the ARU-Net. The proposed method consists of three steps: superpixel calculation, state estimation and superpixel clustering, which are described in the following.

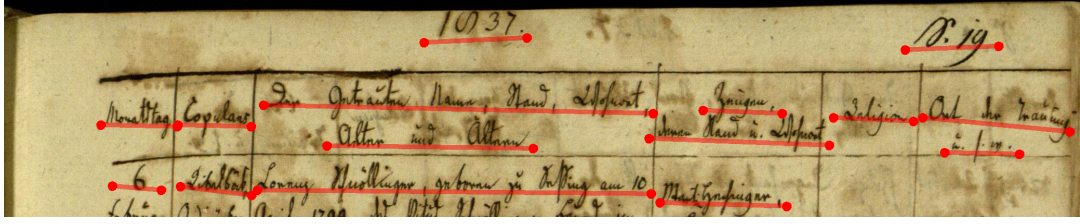
The trained ARU-Net generates an output  $\mathbf{C} \in [0, 1]^{1 \times 2 \times I \times 3}$  for each image  $I \in \mathcal{I}$ . In the following  $B = \mathbf{C}_{:, :, 1} \in \mathcal{I}$  denotes the image encoding the probability of each pixel belonging to a baseline and  $S = \mathbf{C}_{:, :, 2} \in \mathcal{I}$  is the separator image, see Fig.4.11a.

#### Superpixel Calculation

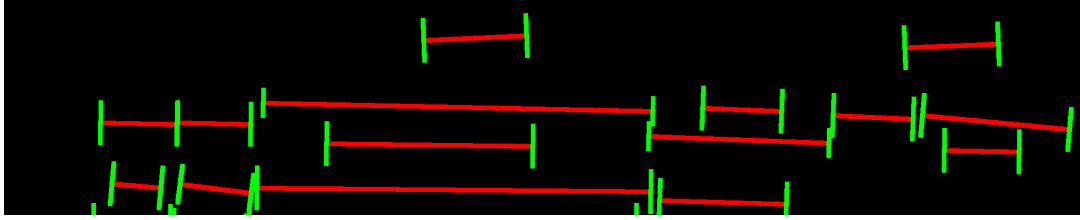
The number of all pixels in an image often exceeds several millions. To reduce the dimensionality of the problem (the number of pixels to be regarded for the baseline estimation), we limit ourselves to a subset of all pixels.

**Definition 4.3.6** (superpixel). Let  $\mathbb{S} = \{\mathbf{p}^{(1)}, \dots, \mathbf{p}^{(N)}\}$  be a subset of the image pixels of  $I$  (typically,  $N \ll {}_1I \cdot {}_2I$  holds). An element of  $\mathbb{S}$  is called *superpixel* (SP).

Basically, the definition of a superpixel does not introduce any new concept. An SP is just a pixel which is somehow regarded to be of certain importance. Because it is a frequently used term, we decided to introduce it via a definition. The set of SPs constitutes the basis for the clustering process to build baselines. Hence, it is easy



(a) **Baseline ground truth** – The baselines are described by the red dots. For visualization purposes the dots of a baseline were connected.



(b) **Pixel ground truth produced by Alg. 6** – Green encodes the separator class, red the baseline class and black the "other" class.

Figure 4.10.: **Baseline and pixel ground truth** – These are shown for the top snippet of the image of Fig. 4.6.

to see that the choice of the set of SPs is crucial for the overall performance. If there are no SPs in  $\mathbb{S}$  which belong to a certain baseline, this baseline will be not detected. To calculate a suitable set of SPs, we utilize the baseline image  $B$  generated by the ARU-Net.

In a first step,  $B$  is binarized

$$B_{i,j}^{(b)} = \begin{cases} 1 & , \text{ if } B_{i,j} > b \\ 0 & , \text{ else} \end{cases} \quad , \quad \forall i \in [{}_1B], j \in [{}_2B],$$

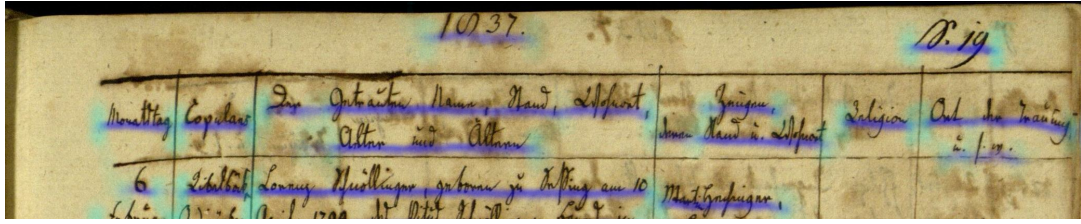
see Fig. 4.11b. The morphological skeleton  $B^{(s)} = \text{SKE}(B^{(b)})$  is calculated for  $B^{(b)}$  following Lantuéjoul's formula [Ser82], see Alg. 7. The skeleton image of  $B^{(b)}$  is depicted in Fig. 4.11c.

All foreground pixels (pixels with an intensity of 1) of  $B^{(s)}$  build an initial set of pixels  $\{\mathbf{p}^{(1)}, \dots, \mathbf{p}^{(M)}\}$ . Its elements are sorted ( $\pi : \mathbb{N} \rightarrow \mathbb{N}$  permutation) in descending order w.r.t. their baseline confidences

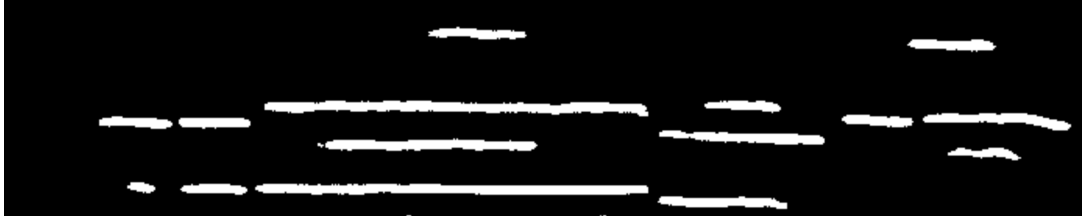
$$(\mathbf{p}^{(\pi(1))}, \dots, \mathbf{p}^{(\pi(M))}) : B_{\mathbf{p}^{(\pi(i))}} \geq B_{\mathbf{p}^{(\pi(j))}} \Leftrightarrow i \leq j. \quad (4.3.1)$$

Finally,  $\mathbb{S}$  is set up by iteratively adding pixels of the sorted list of Eq. (4.3.1) (beginning with the first pixel). To keep the number of SPs small, a new pixel  $\mathbf{p}$  is added to  $\mathbb{S}$  only if

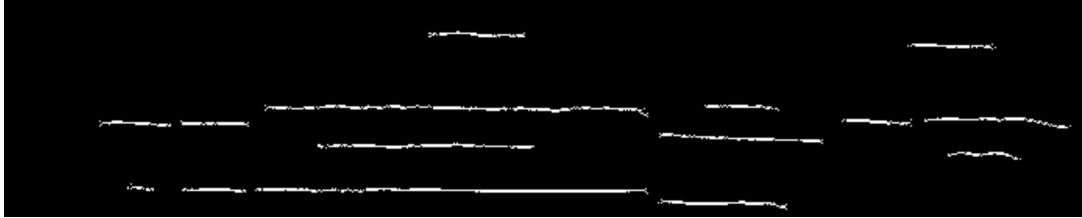
$$\|\mathbf{p} - \mathbf{q}\|_2 > d \quad \forall \mathbf{q} \in \mathbb{S} \quad (4.3.2)$$



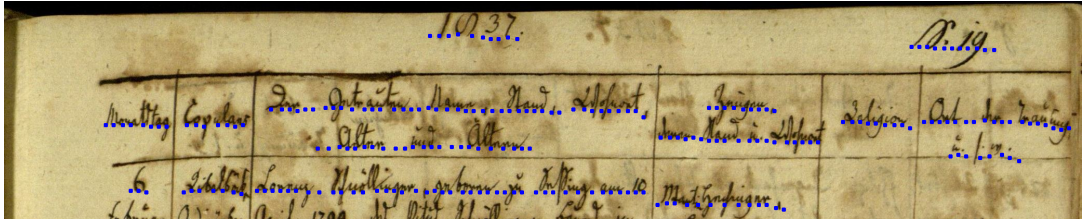
(a) **ARU-Net output** – The estimated baseline image  $B$  (blue) and separator image  $S$  (cyan) are shown as underlay of the original image.



(b) **Binarized baseline image** – Shown is the binarized baseline image  $B^{(b)}$ .



(c) **Skeleton baseline image** – Shown is the skeleton image  $B^{(s)}$  computed by Alg. 7.



(d) **Superpixel** – The calculated set  $\mathbb{S}$  of SPs (blue) is shown.

Figure 4.11.: **Superpixel calculation** – The intermediate steps of the superpixel calculation are shown for the top snippet of the image of Fig. 4.6.

holds for a distance threshold  $d \in \mathbb{R}$ , otherwise it is skipped. In Fig. 4.11d the set of resulting SPs is shown. These SPs build the basis for the further clustering.

**Remark 4.3.7.** For all experiments, we have chosen fixed values of  $b = 0.2$  (binarization threshold) and  $d = 10$  (Eq. (4.3.2)). These demonstrated to be well suited for a wide range of different scenarios. Hence, they are not regarded as free parameters of the system which have to be further tuned. This also holds for the parameters which are fixed in Rem. 4.3.15 & 4.3.24.

---

**Algorithm 7:** Skeleton Calculation (Lantuéjoul's formula):  $I^{(s)} = SKE(I^{(b)})$ 


---

**input** : binary image  $I^{(b)}$   
**output**: skeleton image  $I^{(s)}$   
1  $E \leftarrow$  structuring element  $\triangleright$  e.g., cross of size 3  
2  $I^{(s)} \leftarrow \underline{0}$   $\triangleright$  of dimension  ${}_1I^{(b)} \times {}_2I^{(b)}$   
3 **while**  $I^{(b)} \neq \underline{0}$  **do**  
4      $T \leftarrow \neg(I^{(b)} \circ E)$   $\triangleright \circ$  morphological opening  
5      $T \leftarrow T \wedge I^{(b)}$   
6      $I^{(s)} \leftarrow I^{(s)} \vee T$   
7      $I^{(b)} \leftarrow I^{(b)} \ominus E$   $\triangleright \ominus$  morphological erosion  
**return** :  $I^{(s)}$

---

**Supapixel State Estimation**

Assume we can assign each SP to a certain text line. The state of an SP should encode meaningful characteristics of its text line. These characteristics will be defined and combined to constitute the state of an SP. This work is based on previous work of [RKC14]; [Grü+17], but adapted to the characteristics of SPs extracted given the ARU-Net output, e.g., an easier calculation of the local text orientation is utilized as well as a different smoothing term formulation.

**Definition 4.3.8** (local text orientation). The *local text orientation* ( $\omega_p$ ) of an SP  $p$  is the slope of its text line's baseline at the coordinates closest (w.r.t. the euclidean distance) to  $p$ .

**Definition 4.3.9** (interline distance). The *interline distance* ( $s_p$ ) of an SP  $p$  is the off-text distance of its text line's baseline to the nearest other baseline.

**Definition 4.3.10** (state). The *state* of an SP is the pair  $(\omega_p, s_p)$  of its local text orientation and its interline distance.

In the following, we will describe a method to estimate the states of all SPs. The local text orientation of an SP will be calculated in a straightforward way utilizing solely the baseline image  $B$  along with the position of nearby SPs. On the other hand, the estimation of the interline distances combines local information of the text line's periodicity with the more global assumption that nearby SPs tend to have similar interline distances. For this purpose, let  $\mathbb{S}$  be the set of calculated SPs and  $\mathfrak{N}$  is the neighborhood system (cf. Def. 3.2.2) calculated by Delaunay's triangulation [Del34] for the set of SPs, see Fig. 4.13.

**Definition 4.3.11** (connectivity function). Let  $\lambda(\cdot; p, q) : [0, 1] \rightarrow \mathbb{R}^2$  be the line segment defined in Def. 4.2.1. The function  $\Gamma : \mathfrak{N} \times \mathfrak{I} \rightarrow [0, 1]$  defined as

$$\Gamma(e_{p,q}, I) := \frac{\int_0^1 I(\lambda(\tau; p, q)) d\tau}{\|p - q\|_2} \quad (4.3.3)$$



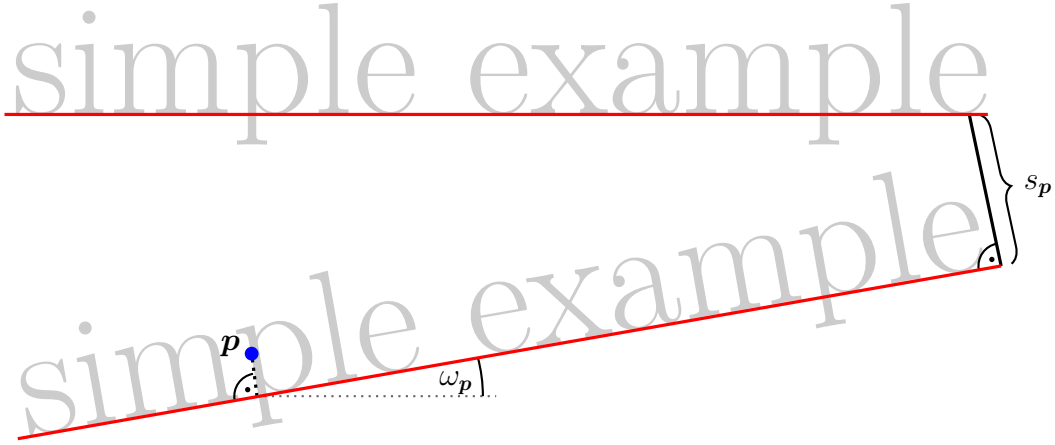


Figure 4.12.: **Local text orientation and interline distance for an SP** – Depicted are the local text orientation and the interline distance for SP  $p$ . Obviously,  $p$  belongs to the lower text line. Consequently its local text orientation ( $\omega_p$ ) is roughly  $10^\circ$ . Its interline distance ( $s_p$ ) is the shortest off-text distance from its textline’s baseline to the adjacent text line.

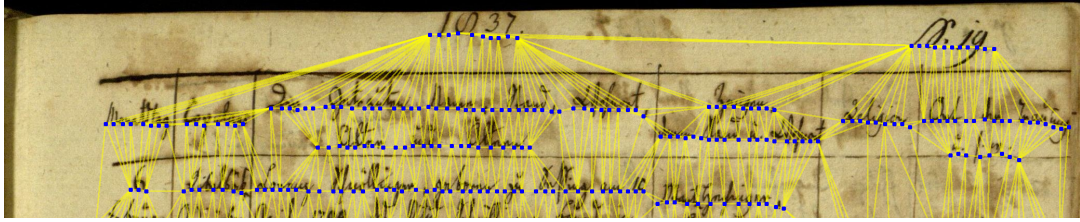


Figure 4.13.: **Delaunay neighborhood system** – Neighborhood system  $\mathfrak{N}$  calculated by Delaunay’s triangulation for the set  $\mathbb{S}$  of SPs of Fig. 4.11d.

is called *connectivity function*.  $I(\lambda(\tau; \mathbf{e}_{p,q}))$  denotes the intensity of the pixel in  $I$  closest (w.r.t the euclidean distance) to the real-valued coordinates  $\lambda(\tau; \mathbf{e}_{p,q})$ , see Rem. 1.3.3.

The connectivity function calculates the average intensity for a given image along the shortest path connecting two pixels. The local text orientation of each SP is estimated as  $\omega_p = \text{LTO}(\mathbf{p}; \mathfrak{N}, B)$  utilizing  $\mathfrak{N}$  and the baseline image  $B$ , see Alg. 8. The LTO algorithm picks the two neighbors of an SP  $p$  with the largest baseline connectivity to  $p$  and determines the slope of the line passing through these neighbors.

The periodicity of text lines in document images is utilized to calculate the interline distances. We determine the interline distance of an SP  $p$  by evaluating the regional text-line periodicity around  $p$  as follows. For an SP  $p$ , for a circular region of diameter  $d \in \mathbb{N}$  around  $p$ , and for a projection direction determined by the local

---

**Algorithm 8:** Local Text Orientation of  $\mathbf{p}$ :  $\omega_{\mathbf{p}} = LTO(\mathbf{p}, \mathfrak{N}, B)$ 


---

**input** : SP  $\mathbf{p}$ , neighborhood system  $\mathfrak{N}$ , baseline image  $B$

**output**: local text orientation  $\omega_{\mathbf{p}}$  of  $\mathbf{p}$

```

1  $\mathbb{M} \leftarrow \{e_{q,r} \in \mathfrak{N} \mid \mathbf{q} = \mathbf{p} \vee \mathbf{r} = \mathbf{p}\}$ 
2  $\mathcal{L} \leftarrow$  sorted list of  $\mathbb{M}$   $\triangleright$  sorted by means of  $\Gamma(e_{q,r}, B)$ 
3 if  $|\mathcal{L}| = 1$  then
4    $e_{q,r} \leftarrow \mathcal{L}_1$   $\triangleright \mathcal{L}_k$  denotes the  $k$ -th element of  $\mathcal{L}$ 
5 else
6    $e_{q,r} \leftarrow (\mathcal{L}_1 \setminus \mathbf{p}, \mathcal{L}_2 \setminus \mathbf{p})$ 
return :  $\omega_{\mathbf{p}} \leftarrow -\arctan\left(\frac{r_y - q_y}{r_x - q_x}\right)$ 

```

---

text orientation  $\omega_{\mathbf{p}}$ , let

$$\mathbf{h}^{p,d} = (\mathbf{h}_1^{p,d}, \dots, \mathbf{h}_d^{p,d})^T \in \mathbb{N}^d$$

be the projection profile with respect to  $\mathbb{S}$ , see Fig. 4.14. For the calculation of  $\mathbf{h}^{p,d}$ , only SPs with a euclidean distance to  $\mathbf{p}$  of less than  $\frac{d}{2}$  are taken into account.

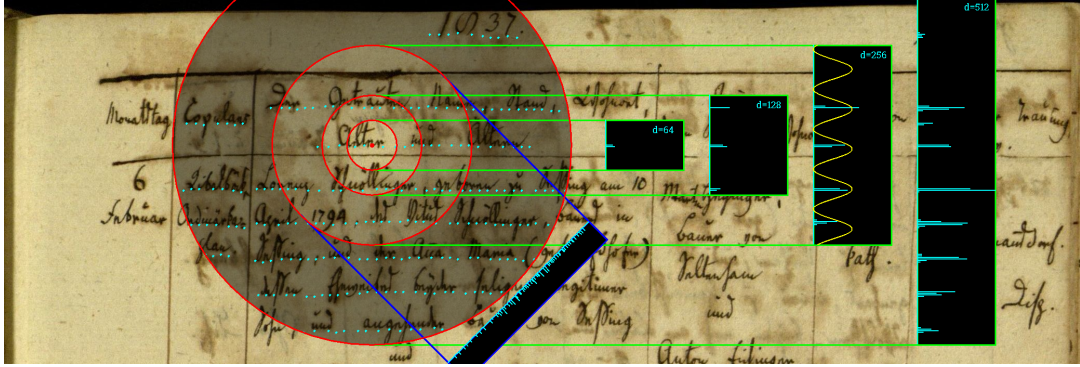


Figure 4.14.: **Interline distance estimation** – Illustration of several projection profiles for a certain SP (red point). The profiles for different diameters  $d \in \{64, 128, 256, 512\}$  and an orientation of  $0^\circ$  are shown in green. The winning period (interline distance) is drawn as yellow curve. In blue a histogram for a wrong orientation ( $45^\circ$ ) is shown.

**Remark 4.3.12.** The projection profile  $\mathbf{h}^{p,d}$  can be calculated very efficiently by utilizing Lem. 4.2.5 for the off-text distance of  $\mathbf{p}$  to  $\mathbf{q} \in \mathbb{S}$  with  $\|\mathbf{p} - \mathbf{q}\|_2 \leq \frac{d}{2}$ . Of course, the sign of the off-distance has to be regarded, cf. Eq. (4.2.5).

To extract the regional periodicity inherent in the projection profile  $\mathbf{h}^{p,d}$ , a *discrete fourier transform (DFT)* [Bri88] is applied to  $\mathbf{h}^{p,d}$  with resulting coefficients

$$\mathcal{H}^{p,d} = (\mathcal{H}_1^{p,d}, \dots, \mathcal{H}_d^{p,d})^T \in \mathbb{C}^d. \quad (4.3.4)$$

A coefficient  $\mathcal{H}_k^{p,d}$ ,  $k \in [d]$  corresponds to the portion of the signal with a period of  $\frac{d}{k}$  to the entire discrete signal  $\mathbf{h}^{p,d}$ .

In the simplest case, the index  $k'$  of the dominant coefficient of  $\mathcal{H}^{p,d}$  determines the interline distance of  $\mathbf{p}$  as  $s_p = \frac{d}{k'}$ . However, we may be forced to assign a different value to  $s_p$  due to additional constraints to be discussed in a moment. Therefore, a data term (cf. Eq. (3.2.1)) is introduced. This data term assigns a negative confidence to an interline spacings determined by  $d$  and  $k$  under the data (the set of SPs).

**Definition 4.3.13** (data term). The *data term* of SP  $\mathbf{p}$  and interline distance  $\frac{d}{k}$  is defined as

$$D\left(\frac{d}{k}; \mathbf{p}\right) := -\log\left(\frac{|\mathcal{H}_k^{p,d}|^2}{\|\mathcal{H}^{p,d}\|_2^2}\right). \quad (4.3.5)$$

Remarkably, the normalization of the DFT coefficients by the quadratic  $L_2$ -norm of  $\mathcal{H}^{p,d}$  reduces the effect of the different diameters  $d$ .

The data term is used within an energy minimization framework for finding the optimal interline distance for each SP, cf. Sec. 3.2. Hence, we aim for an optimal labeling of the set of SPs. The set of possible interline distances constitutes the set of labels to be assigned to the SPs. To cover a suitable range of different interline distances as well as to be robust against disturbances due to close-by text regions of a different style, the projection profiles and DFTs are calculated for different diameters  $d \in \{64, 128, 256, 512\}$  and  $k \in \{3, 4, 5\}$ . The choice of the values for  $d$  and  $k$  is application driven and results in a set of possible interline distances (see Tab. 4.4) which are reasonable concerning typically document resolutions and layouts.

Table 4.4.: **Set of possible interline spacings** – The resulting interline spacings for the chosen projection profile diameters  $d \in \{64, 128, 256, 512\}$  and fourier coefficient indices  $k \in \{3, 4, 5\}$  are shown.

$d/k$	5	4	3
64	12.8	16.0	21.3
128	25.6	32.0	42.7
256	51.2	64.0	85.3
512	102.4	128.0	170.7

Thus, the resulting set of possible labels comprises  $l = 12$  elements. We identify each of the interline spacings with a natural number between 1 and 12 in a sorted fashion, i.e., 12.8 is identified with 1 and 170.7 is identified with 12. For the reason of simplicity, we write  $D(i; \mathbf{p}) \in [l]$  for the data term (cf. Eq. (4.3.5)) of interline

spacing  $i$  assigned to SP  $\mathbf{p}$ . In the following, we aim for a labeling  $\Lambda : \mathbb{S} \rightarrow [l]$  which assigns a label to each SP.

Following a greedy labeling strategy by assigning the interline distance with the lowest data energy defined by Eq. (4.3.5) to each SP, i.e., solving the labeling problem

$$\Lambda^* = \arg \min_{\Lambda \text{ is labeling}} \sum_{\mathbf{p} \in \mathbb{S}} D(\Lambda(\mathbf{p}); \mathbf{p}), \quad (4.3.6)$$

leads to a noisy result, see Fig. 4.15a.

To reduce the noise effects, the influence of close-by SPs is taken into account. It is reasonable to expect that neighboring SPs tend to have similar interline distances. This expectation is encoded via a smoothing term. The smoothing term is equally defined for all adjacent SPs. Consequently, it is not dependent on the respective edge as in Eq. (3.2.1).

**Definition 4.3.14** (smoothing term). Let  $\sigma \in \mathbb{R}_+$  be a penalty value. The *smoothing term* for two labels  $i, j \in [l]$  is defined as

$$V(i, j) := \begin{cases} \sigma & , |i - j| \geq 4 \\ |i - j| & , \text{else} \end{cases}.$$

Thus, the smoothing cost  $V(i, j)$  becomes large if the labels differ. A maximum value of  $\sigma$  is used for "huge" differences. It is easy to see, that the smoothing term is a semi-metric. Because the penalty value is typically large, e.g.,  $\sigma = 25$ , the triangle inequality does not hold:  $25 = V(1, 5) \not\leq V(1, 3) + V(3, 5) = 4$ . Finally, we can formulate the labeling energy like in Eq. (3.2.1). Let  $\Lambda : \mathbb{S} \rightarrow [l]$  be a labeling. The *labeling energy* is defined as

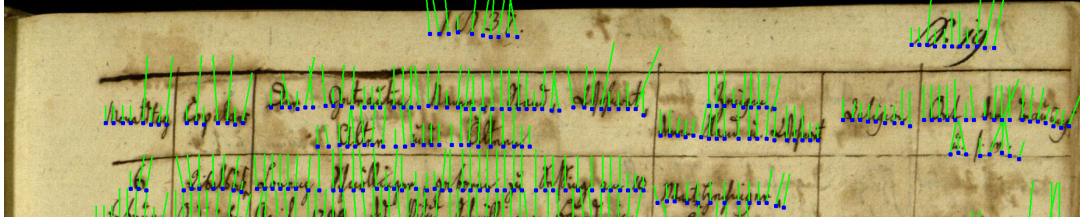
$$E(\Lambda; \mathbb{S}, \mathfrak{N}) = \sum_{\mathbf{p} \in \mathbb{S}} D(\Lambda(\mathbf{p}); \mathbf{p}) + \sum_{\mathbf{e}_{\mathbf{p}, \mathbf{q}} \in \mathfrak{N}} V(\mathbf{p}, \mathbf{q}).$$

The resulting labeling problem

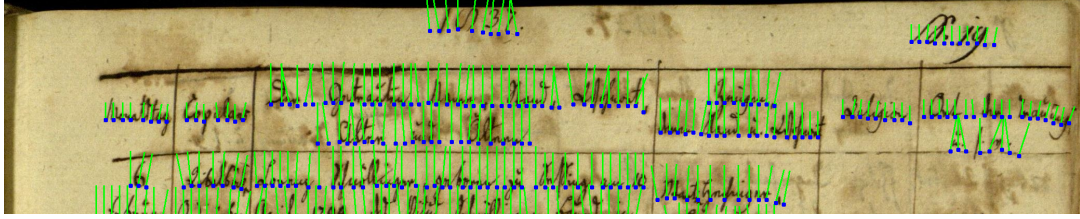
$$\Lambda^* = \arg \min_{\Lambda \text{ is labeling}} E(\Lambda; \mathbb{S}, \mathfrak{N}) \quad (4.3.7)$$

is approximately solved utilizing the energy minimization framework introduced in Sec. 3.2. The additional smoothing term (with a meaningful penalty value  $\sigma$ ) prevents neighboring SPs to differ too much in their interline distances. An exemplary optimized labeling is shown in Fig. 4.15b.

**Remark 4.3.15.** For all experiments, we have chosen a fixed value of  $\sigma = 25$ .



(a) **Greedy states** – The SP states for the greedy labeling (Eq. (4.3.6)) are shown.



(b) **Final states** – The SP states for the smoothed labeling (Eq. (4.3.7)) are shown.

Figure 4.15.: **SPs along with their assigned states** – The local text orientation of each SP is visualized by the orientation of the green lines (for better clarity rotated by  $90^\circ$ ). The length of the lines encode the interline distance of the corresponding SP.

### Supapixel Clustering

In the previous sections, the calculation of SPs and their enrichment with state information was described. In a final step, this state information is utilized to cluster the SPs to build baselines. There will be a one-to-one assignment between clusters and baselines. In the following, we call a subset of  $\mathbb{S}$  *cluster*. Thus, a cluster is just a set of SPs.

In this section, we formulate the clustering problem and introduce a greedy clustering procedure to solve the problem. Two assumptions which hold for baselines in general (independent of the baseline intuition) constitute the conditions for the clustering problem:

- (I) Baselines should not exceed a certain curvilinearity value.
- (II) Within the interline distance of a baseline there are no other baselines.

Basically, assumption (I) claims that a baseline can be approximated by a polynomial function of a certain degree, see [RKC14]. Assumption (II) is self-explanatory.

**Remark 4.3.16.** In the following,  $\omega(\{\mathbf{p}^{(1)}, \dots, \mathbf{p}^{(n)}\})$  denotes the average orientation and  $s(\{\mathbf{p}^{(1)}, \dots, \mathbf{p}^{(n)}\})$  the average interline distance of all SPs in  $\{\mathbf{p}^{(1)}, \dots, \mathbf{p}^{(n)}\}$ .

**Definition 4.3.17** (curvilinearity value). Let  $deg \in \mathbb{N}$  and  $\mathbb{S}^{(1)} \subset \mathbb{S}$  be a cluster. Let  $p_{\mathbb{S}^{(1)}, deg}(t) \in \mathbb{P}[t]$  be the polynomial which solves the linear regression problem

in the monomials  $t^0, t^1, \dots, t^{deg}$  for the rotated real-valued pixels  $\mathbb{S}^{(1)'} defined as$

$$\mathbb{S}^{(1)'} := \left\{ \begin{pmatrix} \cos(-\omega(\mathbb{S}^{(1)})) & -\sin(-\omega(\mathbb{S}^{(1)})) \\ \sin(-\omega(\mathbb{S}^{(1)})) & \cos(-\omega(\mathbb{S}^{(1)})) \end{pmatrix} \cdot \mathbf{p} \mid \mathbf{p} \in \mathbb{S}^{(1)} \right\} \subset \mathbb{R}^2. \quad (4.3.8)$$

The root-mean-square regression error normalized by  $s(\mathbb{S}^{(1)})$  is called *curvilinearity value* of  $\mathbb{S}^{(1)}$  and is denoted by  $cur(\mathbb{S}^{(1)}, deg)$ .

**Remark 4.3.18.** We fix  $deg = 3$  and omit it in the following.

Note that the rotation matrix in Eq. 4.3.8 for computer vision coordinates as introduced in Def. 1.3.1 is equal to the rotation matrix in classical coordinates.

**Lemma 4.3.19.** *The rotation matrix in the computer vision coordinates as defined in Def. 1.3.1 is given by*

$$R_\alpha = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}.$$

*Proof.* The transformation matrices from computer vision coordinates to classical coordinates ( $T$ ) and vice versa ( $T^{-1}$ ) are given by

$$T = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \quad T^{-1} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}.$$

Let  $\tilde{R}_\alpha = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}$  denote the well-known rotation matrix in classical coordinates and  $\mathbf{p}$  is any pixel. The rotation of  $\mathbf{p}$  by  $\alpha$  can now be calculated as

$$\mathbf{p}' = T^{-1} \cdot \tilde{R}_\alpha \cdot T \cdot \mathbf{p}.$$

With the associative law and  $R_\alpha = T^{-1} \cdot \tilde{R}_\alpha \cdot T$  the lemma is proven.  $\square$

The curvilinearity value allows for an easy evaluation of (I). To test for (II) we will introduce the distance of two clusters. Remarkably, only distances orthogonal to the text orientation, the off-text distances as introduced in Def. 4.2.4, should be taken into account. Therefore, the off-text distance is generalized for two clusters of SPs.

Calculating the minimal pairwise off-text distance of all SPs of two clusters could result in a cluster distance distorted by SP outliers. Therefore, SPs in each cluster are projected onto the corresponding regression curve obtained by the regression problem of Def. 4.3.17, before taking pairwise distances.

**Definition 4.3.20** (regression curve). Let  $\mathbb{S}^{(1)}$ ,  $\mathbb{S}^{(1)'}$  and  $p_{\mathbb{S}^{(1)}}(t)$  be of Def. 4.3.17. The *spatial  $t$ -range* of  $\mathbb{S}^{(1)'}$  is given by  $t_{min} = \min\{\mathbf{p}_2 \mid \mathbf{p} \in \mathbb{S}^{(1)'}\}$  and  $t_{max} = \max\{\mathbf{p}_2 \mid \mathbf{p} \in \mathbb{S}^{(1)'}\}$ . A curve  $c_{\mathbb{S}^{(1)}} : [0, 1] \rightarrow \mathbb{R}^2$  which results from rotating the

graph of  $p_{\mathbb{S}^{(1)}}(t)$  for  $t \in [t_{min}, t_{max}]$  by  $\omega(\mathbb{S}^{(1)})$  is called *regression curve* of  $\mathbb{S}^{(1)}$ , see Fig. 4.16.

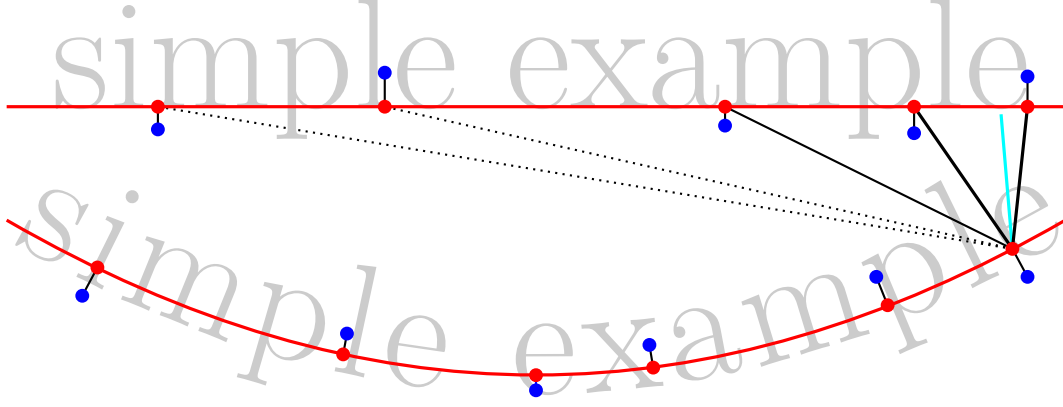


Figure 4.16.: **Regression curve and cluster distance** – Displayed are two clusters  $\mathbb{S}^{(1)}$ ,  $\mathbb{S}^{(2)}$  as blue points. Their corresponding regression curves  $c_{\mathbb{S}^{(1)}}(t)$ ,  $c_{\mathbb{S}^{(2)}}(t)$  are shown as red curves. The projected SPs  $\tilde{\mathbb{S}}^{(1)}$ ,  $\tilde{\mathbb{S}}^{(2)}$  are the red points. For the cluster distance calculation, only pairs of projected SPs of the different clusters for which the euclidean distance is less than a certain value (solid black lines) are taken into account. The resulting cluster distance (off-text distance with respect to the average slope of the regression curve at the respective positions) is depicted as cyan line.

The SPs in  $\mathbb{S}^{(1)}$  are projected (in off-text direction) onto  $c_{\mathbb{S}^{(1)}}$ . The resulting projected SPs are denoted by  $\tilde{\mathbb{S}}^{(1)}$ , see Fig.4.16. To achieve robust distance estimates even for curved and differently slanted text lines we focus on SPs of the different clusters which are quite close to each other and furthermore take into account the slope of the regression curve at the specific SP positions instead of averaging over the entire text line.

**Definition 4.3.21** (cluster distance). Assume two clusters  $\mathbb{S}^{(1)}$ ,  $\mathbb{S}^{(2)}$  with regression curves  $c_{\mathbb{S}^{(1)}}(t)$ ,  $c_{\mathbb{S}^{(2)}}(t)$  and projected SPs  $\tilde{\mathbb{S}}^{(1)}$ ,  $\tilde{\mathbb{S}}^{(2)}$ . The *cluster distance* is defined as

$$d(\mathbb{S}^{(1)}, \mathbb{S}^{(2)}) := \min_{\substack{\mathbf{p} \in \tilde{\mathbb{S}}^{(1)}, \mathbf{q} \in \tilde{\mathbb{S}}^{(2)}: \\ \|\mathbf{p} - \mathbf{q}\|_2 < 4 \cdot s(\mathbb{S}^{(1)} \cup \mathbb{S}^{(2)})}} d_{off}(\mathbf{p}, \mathbf{q}; \omega^c(\mathbf{p}, \mathbf{q})),$$

$\omega^c(\mathbf{p}, \mathbf{q})$  is the average slope of the corresponding regression curves at  $\mathbf{p}$  and  $\mathbf{q}$ , respectively (see Fig. 4.16).

The cluster distance is utilized to evaluate condition (II). Finally, we will use conditions (I) to introduce feasible sets of clusters. For this purpose, we will limit



ourselves to partitions (cf. Def. 3.2.3) of  $\mathbb{S}$ . Hence, each SP has to be assigned to one baseline at most. Furthermore, we require the baseline clusters to be  $\mathfrak{N}$ -linked.

**Definition 4.3.22** ( $\mathfrak{N}$ -linked). Let  $\mathbb{S}^{(1)}$  be a cluster and  $\mathfrak{N}$  be a neighborhood system.  $\mathbb{S}^{(1)}$  is  $\mathfrak{N}$ -linked iff

$$\forall \mathbf{p}, \mathbf{q} \in \mathbb{S}^{(1)} \exists \mathbf{p}_0, \dots, \mathbf{p}_N \in \mathbb{S}^{(1)} : \mathbf{p}_0 = \mathbf{p} \wedge \mathbf{p}_N = \mathbf{q} \wedge \mathbf{e}_{\mathbf{p}_i, \mathbf{p}_{i+1}} \in \mathfrak{N} \quad (0 \leq i \leq N-1)$$

holds.

**Definition 4.3.23** (feasible). For  $\gamma, \delta \in \mathbb{R}_+$ , a set of SPs  $\mathbb{S}$  and a neighborhood system  $\mathfrak{N}$ , we call a partition  $\mathcal{P} = \{\mathbb{S}^{(0)}, \dots, \mathbb{S}^{(L)}\}$  of  $\mathbb{S}$  with  $L \in \mathbb{N}$  *feasible* iff

- $\mathbb{S}^{(i)}$  is  $\mathfrak{N}$ -linked  $\forall i > 0$
- conditions (I) and (II) hold, i.e.,
  - $\text{cur}(\mathbb{S}^{(i)}) < \gamma \quad \forall i > 0$
  - $d(\mathbb{S}^{(i)}, \mathbb{S}^{(j)}) > \delta \cdot \max\{s(\mathbb{S}^{(i)}), s(\mathbb{S}^{(j)})\} \quad \forall i, j > 0, i \neq j.$

The set of feasible partitions is denoted by  $\text{feas}_{\mathfrak{N}}(\mathbb{S}; \gamma, \delta)$ .

Note that the number  $L \in \mathbb{N}$  of baselines is (a-priori) unknown. We usually omit  $\gamma$  and  $\delta$  and write  $\text{feas}_{\mathfrak{N}}(\mathbb{S})$  for the set of feasible partitions. Let  $\mathcal{P} = \{\mathbb{S}^{(0)}, \dots, \mathbb{S}^{(L)}\}$  be a feasible partition. We follow the convention that the clusters  $\mathbb{S}^{(i)}$ ,  $i > 0$  identify the baselines and  $\mathbb{S}^{(0)}$  constitutes the clutter cluster containing SPs not belonging to any baseline. Finally, we represent the baseline corresponding to  $\mathbb{S}^{(i)}$  by the polygonal chain which is defined by the projected SPs  $\tilde{\mathbb{S}}^{(i)}$ , see Fig. 4.16 & 4.17.

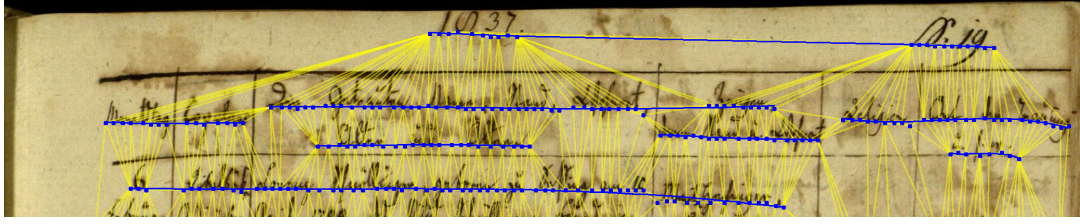
In the following, we will incorporate domain knowledge to promote SPs belonging to different baselines not to be  $\mathfrak{N}$ -linked. Hence, partitions with erroneously connected baselines are not feasible anymore. This is done by a modification of the neighborhood system  $\mathfrak{N}$ . Since baselines of different text orientations should not contribute to the same cluster, we adjust the initial neighborhood system  $\mathfrak{N}$  by removing edges  $\mathbf{e}_{\mathbf{p}, \mathbf{q}}$  of SPs with substantially different local text orientations:  $|\omega_{\mathbf{p}} - \omega_{\mathbf{q}}| \bmod \pi > \frac{\pi}{4}$ .

In addition, it is an ease to incorporate layout information by further adjusting  $\mathfrak{N}$ . The layout information which is encoded by the separator image  $S$  (Fig. 4.11a) can be incorporated by taking into account the connectivity of the SPs in  $S$ . Edges  $\mathbf{e}_{\mathbf{p}, \mathbf{q}} \in \mathfrak{N}$  for which a separator is crossed, i.e.,  $\Gamma(\mathbf{e}_{\mathbf{p}, \mathbf{q}}, S) > \eta$  or  $\max_{\tau \in [0, 1]} S(\lambda(\tau; \mathbf{e}_{\mathbf{p}, \mathbf{q}})) > 2 \cdot \eta$  ( $\Gamma$  of Def. 4.3.11 and  $\lambda$  of Def. 4.2.1) holds, are removed, see Fig. 4.17b.

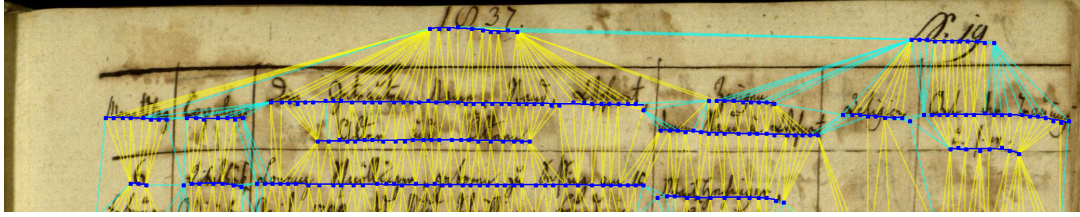
Finally, a common scenario is the baseline detection with given text regions. We assume that the text regions are represented by closed polygonal chains  $\mathcal{R}^{(1)}, \dots, \mathcal{R}^{(N)}$ . This additional layout information (if available) is easy to integrate by

$$\mathfrak{N}' = \{\mathbf{e}_{\mathbf{p}, \mathbf{q}} \in \mathfrak{N} \mid \exists i \in [N] : \mathbf{p}, \mathbf{q} \in \mathcal{R}^{(i)}\}.$$





(a) **Without separator information** – The entire neighborhood system (yellow) is shown.



(b) **With separator information** – The neighborhood system was reduced by removing edges (cyan) with high separator connectivity. The corresponding separator information is illustrated in Fig. 4.11a.

Figure 4.17.: **Influence of the separator information on the clustering** – The resulting baselines (blue lines) with and without taking into account the separator information are shown.

Roughly speaking, a closed polygonal chain *contains* a SP if for all "ways" from the SP to the image border one has to cross the polygonal chain. Hence, SPs which are part of different non-overlapping text regions are not  $\mathfrak{N}$ -linked any more. Thus, each baseline  $\mathbb{S}^{(i)}$ ,  $i > 0$  is entirely contained in one text region for all feasible sets. The resulting neighborhood system is still denoted by  $\mathfrak{N}$  (instead of  $\mathfrak{N}'$ ).

**Remark 4.3.24.** For all experiments, we have chosen fixed values of  $\gamma = 0.3$ ,  $\delta = 0.5$  (Def. 4.3.23) and  $\eta = 0.125$ .

After reducing the neighborhood system, we now introduce the total baseline energy. We will assign an energy to all feasible sets and aim for an optimal one. This allows for the formulation of the clustering problem to be solved.

**Definition 4.3.25** (total baseline energy). Let  $B$  be a baseline image,  $\mathfrak{N}$  a neighborhood system and  $\mathcal{P} = \{\mathbb{S}^{(0)}, \dots, \mathbb{S}^{(L)}\}$  a partition over  $\mathbb{S}$ . With  $\mathfrak{N}(\mathbb{S}^{(i)}) = \{e_{p,q} \in \mathfrak{N} \mid p, q \in \mathbb{S}^{(i)}\} \subset \mathfrak{N}$  the *total baseline energy* is defined as

$$b(\mathcal{P}) := \sum_{i=1}^L \sum_{e_{p,q} \in \mathfrak{N}(\mathbb{S}^{(i)})} \Gamma(e_{p,q}, B). \quad (4.3.9)$$

Finally, the clustering problem can be formulated as

$$\mathcal{P}^* = \arg \max_{\mathcal{P} \in \text{feas}_{\mathfrak{N}}(\mathbb{S})} b(\mathcal{P}). \quad (4.3.10)$$

Because there is usually a huge number of feasible partitions, we introduce a greedy clustering algorithm to solve Eq. (4.3.10). The proposed algorithm clusters edges of  $\mathfrak{N}$  instead of clustering SPs. If an edge is assigned to a cluster (set) of edges, we assign both corresponding SPs to the corresponding cluster of SPs. In a first step, the set of edges in  $\mathfrak{N}$  is sorted in decreasing order w.r.t.

$$\left(1 - \frac{d_{\text{off}}(\mathbf{p}, \mathbf{q}; \omega(\{\mathbf{p}, \mathbf{q}\}))}{\|\mathbf{p} - \mathbf{q}\|_2}\right) \cdot \Gamma(\mathbf{e}_{\mathbf{p}, \mathbf{q}}, B). \quad (4.3.11)$$

The sorted list is denoted by  $\mathcal{N}$ . Eq. (4.3.11) takes into account the  $B$ -connectivity value of an edge and discounts it if  $\mathbf{e}_{\mathbf{p}, \mathbf{q}}$  is rather orthogonal to  $\omega(\{\mathbf{p}, \mathbf{q}\})$ . Discounted edges are less likely part of a baseline and are therefore sorted to the end of the list. This avoids that these edges are falsely assigned to baseline clusters which are composed of just a few correct edges (statistics of the cluster are not reliable, yet). Given  $\mathbb{S}$  and  $\mathcal{N}$ , the proposed clustering process is shown in Alg. 9.

**Lemma 4.3.26.** *Alg. 9 converges and yields a feasible partition.*

*Proof.* The while-loop stops if no edge was clustered. Hence, in the worst-case a single edge is clustered each iteration. Consequently, the algorithm stops after (at most)  $|\mathcal{N}|$  while iterations. In practice, it usually converges after less than 10 iterations.

It is easy to see that  $\mathcal{P}$  is a partition. In the beginning, all SPs are in  $\mathbb{S}^{(0)}$ . If SPs are removed from any cluster they are put to exactly one other cluster, cf. line 9, 14, 17 & 22. As a result, each SP is assigned to exactly one cluster during the entire clustering process. Hence,  $\mathcal{P}$  is always a partition.

Each  $\mathbb{S}^{(i)}$ ,  $\forall i > 0$  is  $\mathfrak{N}$ -linked. This is due to the fact, that edges are clustered instead of SPs. I.e., if a new cluster is created, it is  $\mathfrak{N}$ -linked, cf. line 10. If it is extended, the new SP is connected to the cluster via an edge, cf. line 14. If two clusters are merged they are both  $\mathfrak{N}$ -linked and connected via an edge, cf. line 17. Consequently, the merged cluster is also  $\mathfrak{N}$ -linked.

The curvilinearity is explicitly checked before a cluster is changed, cf. line 12 & 16. Of course, a cluster consisting of two SPs (line 10) has a curvilinearity value of 0.

The for-loop (line 19) ensures that the distance condition holds. As a result, the resulting partition is feasible.  $\square$

## 4.4. Experiments

The experiment section is divided into 4 parts. First, each method relying on supervised (deep) learning and therefore relying on training data can suffer from the need of an enormous amount of labeled training data. We demonstrate that the presented

**Algorithm 9:** SP Clustering:  $\mathcal{P} = SPC(\mathbb{S}, \mathcal{N})$ 


---

**input** : set of SPs  $\mathbb{S}$  and sorted list of edges  $\mathcal{N}$   $\triangleright \gamma, \delta$  of Rem. 4.3.24  
**output**: optimized partition  $\mathcal{P}$

```

1  $\mathbb{S}^{(0)} \leftarrow \mathbb{S}, \mathcal{P} \leftarrow \{\mathbb{S}^{(0)}\}, n \leftarrow 0$ 
2 while  $|\mathcal{N}| \neq n$  do
3    $n \leftarrow |\mathcal{N}|$ 
4   for  $e_{p,q} \in \mathcal{N}$  do  $\triangleright$  Four possible cases dependent on the SPs
5     if  $\exists i > 0 : p, q \in \mathbb{S}^{(i)}$  then  $\triangleright$  Case 1: add edge to cluster!
6        $\mathcal{N} \leftarrow \mathcal{N} \setminus \{e_{p,q}\}$ 
7     else if  $p, q \in \mathbb{S}^{(0)}$  then  $\triangleright$  Case 2: create new cluster?
8       if  $d_{off}(p, q; \omega(\{p, q\})) < \delta \cdot s(\{p, q\})$  then
9          $\mathcal{N} \leftarrow \mathcal{N} \setminus \{e_{p,q}\}, \mathbb{S}^{(0)} \leftarrow \mathbb{S}^{(0)} \setminus \{p, q\}$ 
10         $\mathcal{P} \leftarrow \mathcal{P} \cup \{\{p, q\}\}$ 
11      else if w.l.o.g.  $p \in \mathbb{S}^{(0)} \wedge \exists i > 0 : q \in \mathbb{S}^{(i)}$  then  $\triangleright$  Case 3: extend?
12        if  $cur(\mathbb{S}^{(i)} \cup \{p\}) < \gamma \wedge d(\mathbb{S}^{(i)}, \{p\}) < \delta \cdot s(\mathbb{S}^{(i)})$  then
13          if  $d(\mathbb{S}^{(i)} \cup \{p\}, \mathbb{S}^{(j)}) > \delta \cdot s(\mathbb{S}^{(j)}) \forall j \neq i, j > 0$  then
14             $\mathcal{N} \leftarrow \mathcal{N} \setminus \{e_{p,q}\}, \mathbb{S}^{(i)} \leftarrow \mathbb{S}^{(i)} \cup \{p\}, \mathbb{S}^{(0)} \leftarrow \mathbb{S}^{(0)} \setminus \{p\}$ 
15          else if  $\exists i, j > 0 (i \neq j) : p \in \mathbb{S}^{(i)} \wedge q \in \mathbb{S}^{(j)}$  then  $\triangleright$  Case 4: merge?
16            if  $cur(\mathbb{S}^{(i)} \cup \mathbb{S}^{(j)}) < \gamma \wedge d(\mathbb{S}^{(i)}, \mathbb{S}^{(j)}) < \delta \cdot \min(s(\mathbb{S}^{(i)}), s(\mathbb{S}^{(j)}))$  then
17               $\mathcal{N} \leftarrow \mathcal{N} \setminus \{e_{p,q}\}, \mathbb{S}^{(i)} \leftarrow \mathbb{S}^{(i)} \cup \mathbb{S}^{(j)}$ 
18               $\mathcal{P} \leftarrow \mathcal{P} \setminus \{\mathbb{S}^{(j)}\}$ 
19 for  $\mathbb{S}^{(i)} \in \mathcal{P}, i > 0$  do  $\triangleright$  Remove close cluster
20   if  $\exists \mathbb{S}^{(j)} \in \mathcal{P}, j > 0, j \neq i : d(\mathbb{S}^{(i)}, \mathbb{S}^{(j)}) < \delta \cdot \max(s(\mathbb{S}^{(i)}), s(\mathbb{S}^{(j)}))$  then
21     if  $b(\mathbb{S}^{(j)}) > b(\mathbb{S}^{(i)})$  then  $\triangleright b$  is baseline energy, cf. Eq. (4.3.9)
22        $\mathcal{P} \leftarrow \mathcal{P} \setminus \{\mathbb{S}^{(i)}\}, \mathbb{S}^{(0)} \leftarrow \mathbb{S}^{(0)} \cup \mathbb{S}^{(i)}$ 

```

**return** :  $\mathcal{P}$

---

approach achieves high quality results on the Bozen dataset [Sán+16] with less than 50 full-page training samples by using data augmentation strategies. Along with an annotating effort of just a few minutes per page the adaptation of the proposed method is easy and cheap.

Second, we demonstrate the applicability of the proposed method for images with arbitrarily oriented as well as curved text lines by achieving nearly as good results as for straight  $0^\circ$  oriented text lines.

The third part presents and compares results of different versions of our proposed neural pixel labeler architectures on the very heterogeneous and challenging cBAD dataset [Grü+18a]; [Die+17b]. We perform tests to show the statistical significance of the stated conclusion – the superiority of the proposed ARU-Net in a two-stage

workflow over other architectures and a single-stage workflow.

Finally, we show that the presented approach outperforms state-of-the-art methods on the datasets of 3 recently hosted competitions. For the first competition the proposed similarity score for the origin point detection is utilized. For the other two competitions, we will make use of the similarity score introduced in Sec. 4.2 (F-value) to measure the quality of the baseline detection.

The configuration for all experiments including the hyperparameters of the network architecture as well as the training are summarized in Tab. 4.5. This configuration is the result of an extensive search in the hyperparameter space and results in impressive results for various scenarios/datasets.

Table 4.5.: **Hyperparameters** – The architecture and training configuration which were used in this work are described.

<b>Image pre-processing:</b> input image $I$ is downscaled by a factor of 2 for $\max\{I_1, I_2\} < 2000$ , 3 for $2000 \leq \max\{I_1, I_2\} < 4800$ or 4 followed by a normalization to mean 0 and variance 1 (on pixel intensity level)
<b>RU-Net architecture, see Fig. 4.7 &amp; 4.8:</b> number of scale spaces: 6, initial feature depth: 8, residual depth (activated layers in a residual block): 3, feature increasing and spatial decreasing factor: 2, activation function: ReLu, kernel size: $3 \times 3$ , stride: 1
<b>A-Net architecture:</b> 4 layer CNN, activation function: ReLu, kernel size: $4 \times 4$ , stride: 1, maxpooling of size $2 \times 2$ after each convolution, feature number: 12, 16, 32, 1
<b>ARU-Net architecture, see Fig. 4.9:</b> number of image scales: 5, classifier: $4 \times 4$ convolution layer with softmax activation
<b>Training:</b> weight initialization: Xavier, optimizer: RMSprop, learning rate: 0.001, learning rate decay per epoch: 0.985, weight decay on the $L_2$ norm: 0.0005, exponential moving average on the model weights: 0.9995, mini batch size: 1 (due to memory limitations of the GPU), early stopping: none (trained for a fixed number of epochs)

Since no early stopping based on the loss for any validation set is used, we train on the entire training set. We used Google’s deep learning library Tensorflow [Aba+15] to implement and train the ARU-Net. The SP extraction, state estimation, and clustering are written in Java. The ARU-Net workflow for training and inference as well as a trained network are freely available<sup>2</sup>. The ARU-Net training takes 3 h to 24 h from scratch (dependent on the number of epochs and samples per epoch) on a Titan X GPU. The inference time per image ranges from 2 s to 12 s per image on a dual core laptop (Intel Core i7-6600U with 16 GiB RAM), this reduces to 0.5 s to

<sup>2</sup><https://github.com/TobiasGruening/ARU-Net>

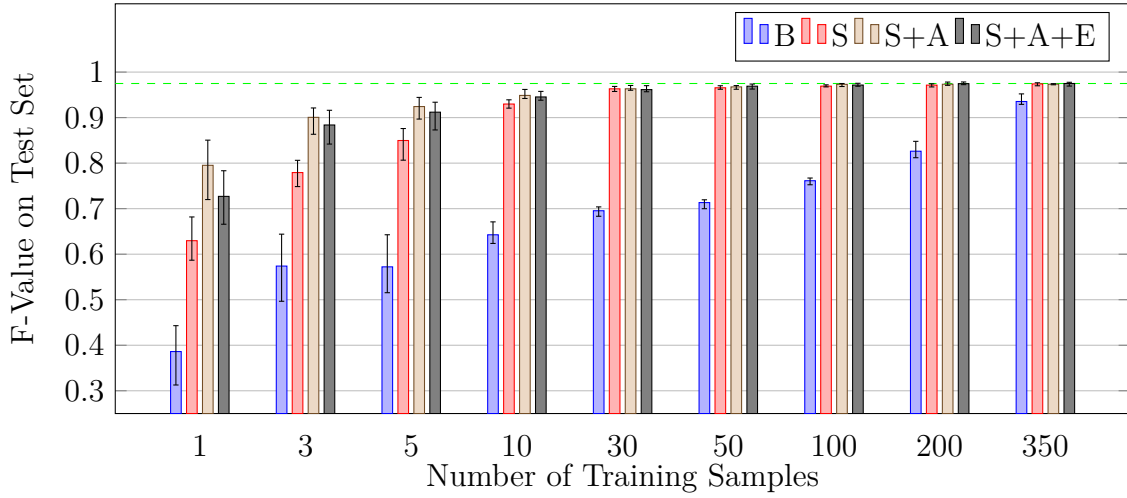


Figure 4.18.: **Influence of the number of training samples and of different data augmentation strategies** – The bar height represents the mean F-value. The error bars encode min-max values of the 5 experiments (not the standard deviation). The dashed green line marks the maximum mean value of 0.975 achieved for 350 trainings samples. For a detailed description of the different augmentation strategies: B (no augmentation), S (arbitrary scaling), S+A (S+ affine transformations) and S+A+E (S+A+ elastic transformations), see main text.

2s running the ARU-Net on the Titan X.

#### 4.4.1. Influence of Training Sample Number and Data Augmentation

A major drawback of state-of-the-art approaches (Sec. 2) is the need for an extensive expert tuning if confronted with scenarios which are not already covered. But the eligibility for a usage at industrial scale depends on the possibility to easily adapt at reasonable cost. For approaches relying on machine learning, this reduces to two questions:

- What about the amount of ground truth needed?
- What about the effort of ground truth production?

Concerning the second question, we refer to Alg. 6. The annotation of baselines for a document image is quite easy and does not need remarkable expert knowledge compared to, e.g., ground truth production for ATR systems for historical handwritings or even the text line annotation at surrounding polygon level. The effort is reduced to several minutes per page by using platforms such as Transkribus<sup>3</sup>. In

<sup>3</sup><https://transkribus.eu>

the following, we want to examine the first question.

The influence of training dataset size along with different data augmentation strategies is investigated for the freely available Bozen dataset<sup>4</sup> [Sán+16], see Fig. A.2. This dataset is a subset of documents from the Ratsprotokolle collection of Bozen composed of minutes of the council meetings held from 1470 to 1805 and consists of 400 pages. Baseline ground truth is available in form of PAGE<sup>5</sup> XML. The dataset is quite challenging concerning layout analysis issues. Most of the pages consist of a single main text region with many difficulties for line detection and extraction, e.g., bleed through, touching text lines and marginalia. For the following experiments, we have randomly divided the Bozen set in a set of training samples  $\mathfrak{T}'$  of size 350 and a test set  $\mathfrak{T}$  of size 50. In a first step, we randomly set up a chain of subsets of  $\mathfrak{T}'$

$$\mathfrak{T}'_1 \subset \mathfrak{T}'_3 \subset \mathfrak{T}'_5 \subset \mathfrak{T}'_{10} \subset \mathfrak{T}'_{30} \subset \mathfrak{T}'_{50} \subset \mathfrak{T}'_{100} \subset \mathfrak{T}'_{200} \subset \mathfrak{T}'_{350},$$

where  $\mathfrak{T}'_i$  contains  $i$  training samples (pages and pixel ground truth). Since we expect an influence of the choice of training samples, i.e., of the sorting of  $\mathfrak{T}'$ , we repeat the mentioned procedure 4 times with different sortings of  $\mathfrak{T}'$ . Notably, the test set remains untouched. Finally, we got 45 training sets – five of each quantity. For each set, we trained the RU-Net for 100 epochs with 256 images per epoch, see Alg. 1. The RU-Net was chosen instead of the ARU-Net, because of the homogeneity of the Bozen dataset concerning font size and resolution. Note that we randomly choose samples of the training set and remove them from the set. If each element of the training set was used for training once, we start again with the initial training set. Hence, it does not matter whether the number of training samples per epoch exceeds the size of the training set or not. This procedure guarantees the same amount of training samples shown to the networks in training independent of the size of the training set. We trained the RU-Net from scratch on all 45 sets in 4 different scenarios. For training purposes the image pre-processing mentioned in Tab. 4.5 is disabled. Instead, the training samples  $(I, \mathbf{G}^{(I)})_i$  are pre-processed following one of the four strategies:

- (a) subsampled by a constant factor of 3 (no further data augmentation - one training sample per element of the training set), referred to as B
- (b) randomly subsampled by a factor  $s \in [2, 5]$ , referred to as S
- (c) S + random affine transformation (three corner points of the image are uniform randomly shifted within a circle of diameter  $0.025 \cdot \max({}_1I, {}_2I)$  around there original position), referred to as S + A
- (d) S + A + elastic transformation [SSP03], referred to as S + A + E

For the test set the images were subsampled by the constant factor of 3 in all scenarios. The results of these 180 experiments are shown in Fig. 4.18.

<sup>4</sup><https://zenodo.org/record/218236>

<sup>5</sup><http://www.primaresearch.org/tools>

One can see that all 3 data augmentation strategies significantly improve the performance compared to the base (B) strategy. Notably, for small numbers of training samples the min-max difference is much larger than for higher number of training samples. Hence, if just a few training samples are available, the choice of these is of large importance. The best mean F-value (0.975) is achieved for all 350 training samples with the S + A + E strategy. Nevertheless, there only is a negligible loss in performance for 200 or 100 training samples. Even for 30 training samples, a F-value of 0.963 is achieved for the S+A strategy, which is sufficient for most applications, see Fig. A.2. This results in a quite acceptable effort for ground truth production making the presented approach interesting even for industrial production. The S + A data augmentation strategy will be the default for the rest of this work.

Of course, the presented numbers are not directly transferable to collections with pages of entirely different scenarios, e.g., census tables mixed with postal cards mixed with ... . One would expect that more than 30 training samples are necessary for this kind of scenario. Nevertheless, the presented experiment reflects a common situation: One has a robust baseline detector which was trained on very heterogeneous data (see Sec. 4.4.4), but this detector does not work satisfyingly well for a certain (in most cases quite homogeneous) collection. The numbers presented here give a hint concerning the effort of ground truth production necessary in this scenario.

#### 4.4.2. Curved and Oriented Text Lines

In this part, we demonstrate the ability of the introduced approach to handle curved or arbitrarily oriented text lines. In a first experiment, the test set of the Bozen dataset was deformed to contain arbitrarily curved text lines. For this purpose, we utilized trigonometric functions with random period to simulate curved text lines in the test phase. The RU-Net was trained (5 times) for 100 epochs with 256 samples per epoch on the Bozen training set using the S + A + E augmentation strategy with strong elastic deformations. We choose elastic transformations in training, because they simulate curves of different amplitudes and frequencies in the same image. Furthermore, we increased the polynomial degree (Def. 4.3.17) to  $deg = 5$  to enable the system to handle the curvatures present in the test set.

**Remark 4.4.1.** Different methods were used to deform the images during training (S + A + E) and test (trigonometric functions) phases. Hence, the system had to learn the concept of curved text lines instead of an inversion of the image degradation method used in the training phase.

In a second experiment, we have trained an RU-Net (5 times) on arbitrarily oriented samples of the Bozen training set and evaluated the resulting networks on oriented pages of the test set. The results are shown in Tab. 4.6 and a few sample images are shown in Fig. A.3.

Table 4.6.: **Results for the Bozen test set** – The results in the Base, Curved and Oriented scenario are depicted. The bold numbers are the averages of 5 experiments. The P- and R-values are strongly related to the well-known precision and recall measures, see Sec. 4.2. Finally, the results for a system trained with all degradations are shown.

Scenario	$\varnothing$ <b>P-Val</b>	$\varnothing$ <b>R-Val</b>	$\varnothing$ <b>F-val</b> [min,max]
Base	<b>0.9765</b>	<b>0.9734</b>	<b>0.9750</b> [0.9693, 0.9770]
Curved	<b>0.9802</b>	<b>0.9690</b>	<b>0.9745</b> [0.9725, 0.9760]
Oriented	<b>0.9625</b>	<b>0.9655</b>	<b>0.9640</b> [0.9582, 0.9674]
	$\varnothing$ <b>F-val</b> (Base)	$\varnothing$ <b>F-val</b> (Curved)	$\varnothing$ <b>F-val</b> (Oriented)
Combined	<b>0.9531</b>	<b>0.9573</b>	<b>0.9676</b>

For the curved scenario the results are as good as for the base scenario. In case of the oriented scenario the results are slightly worse, but still excellent. This demonstrates the applicability for images with curved or oriented text lines without remarkable adaptation of the workflow. Finally, we have trained five models with all degradations (affine, elastic, rotation) and evaluated this model on the three different scenarios. The corresponding F-values are depicted in Tab. 4.6. The system is worse than the experts for the base and curved scenarios, but for the oriented scenario it even benefits from the additional elastic transformations.

#### 4.4.3. U-Net vs. ARU-Net vs. Single-Stage Workflow

In Sec. 4.3, we have introduced the ARU-Net in a two-stage workflow. In this section, we will investigate its superiority over the classical U-Net as well as over a "single-stage" workflow. For this purpose we have trained the U-, RU-, and ARU-Net (each 5 times – random weight initialization and random training sample order) on the recently introduced cBAD dataset<sup>6</sup> [Die+17b]. The details of the dataset are described in [Grü+18a]. In our opinion, this is the most challenging freely available dataset at the moment. We have trained each network for 250 epochs, 1024 training samples each epoch using the S + A data augmentation strategy. To assure the statistical significance of the posed superiority of the newly introduced architecture, we follow [Pui17] and provide the results of a statistical analysis. The choice of appropriate statistical tests is quite limited since we can not make any assumptions regarding the underlying distribution. We utilize 95% confidence intervals (CI) provided by non-parametric bootstrapping [Efr87] as well as the Tukey-Duckworth test (level of significance: 5%) [Tuk59]. The results obtained are summarized in Tab. 4.7.

<sup>6</sup><https://zenodo.org/record/257972>



Table 4.7.: **Results for the cBAD test set** – The results for different neural network architectures and the workflow without Stage II (for the ARU-Net) are shown. Each architecture is trained 5 times on the cBAD train set. The results are sorted with respect to computational effort. The last two columns indicate whether an architecture is superior to all before mentioned ones in terms of disjunct confidence intervals and the Tukey-Duckworth test.

Method	$\emptyset$ <b>F-val</b> [95% CI]		CI	T-D
	Simple Track	Complex Track		
ARU I <sup>†</sup>	<b>0.9627</b> [0.9615, 0.9636]	<b>0.9081</b> [0.9071, 0.9095]		
U	<b>0.9714</b> [0.9701, 0.9721]	<b>0.9114</b> [0.9107, 0.9122]	✓	✓
RU	<b>0.9756</b> [0.9744, 0.9766]	<b>0.9182</b> [0.9165, 0.9203]	✓	✓
ARU	<b>0.9781</b> [0.9772, 0.9789]	<b>0.9223</b> [0.9214, 0.9230]	✓	✓

<sup>†</sup> single-stage workflow – baseline estimation by basic image processing methods (binarization of  $B$  followed by a CC analysis, no usage of  $S$ )

The ARU-Net performs significantly (last two columns) better than all architectures with less computational effort. Furthermore, the results show that the introduction of the second stage is beneficial for the overall performance. Hence, the ARU-Net together with the two-stage workflow has shown its superiority (which is statistically significant) over the other systems and is used in the following. It has to be mentioned that the above comparison is not fair concerning the number of trainable parameters – U - 2.16, RU - 4.13, ARU - 4.14 (in millions) – nor concerning the training and inference time. The comparison is about different architectures which, theoretically, have different capabilities, and whether they make good use of them or not.

#### 4.4.4. Comparison against the State of the Art

In this section, we compare the proposed framework against the state of the art. We have chosen the 3 most recent competitions on text line detection for historical documents, namely:

- ICDAR2015 competition on text line detection in historical documents [Mur+15],
- ICDAR2017 Competition on Layout Analysis for Challenging Medieval Manuscripts (Task 2) [Sim+17],
- cBAD: ICDAR2017 Competition on Baseline Detection [Die+17a].

We will not further introduce the datasets or metrics used and refer to the competition papers.

### ICDAR 2015 Competition on Text Line Detection in Historical Documents (ANDAR-TL)

The ARU-Net was trained on the cBAD training set<sup>7</sup>. This competition aims at the origin point (OP) detection. An OP is roughly spoken the lower left "corner" of a text line. Hence, we calculate the left most point of each detected baseline. This is the output of our system for this competition. The achieved results are shown in Tab. 4.8.

Table 4.8.: **Origing Point (OP) detection results for the ANDAR-TL test set** – Results for the dataset of [Mur+15] are shown. #DF means the number of detection failures (no OP detected by the system), #DM means the number of detection misses (detected OP far away from the ground truth OP) and #FP means the number of false positives.

Method	#HYP	#COR	#DF	#DM	#FP	avg. cost
UNIFR	9301	2578	3022	6456	267	19.00
IA-2	11789	5655	407	6032	102	14.51
A2iA-3 <sup>†</sup>	8967	6523	2490	2263	181	13.20
SNU[Ahn+17]	10466	7741	948	2700	25	9.77
[Grü+17]	10896	8015	517	2860	21	8.19
proposed	11635	9610	358	1942	83	<u>5.39</u>

<sup>†</sup> According to [EGO17] this is an extension of [Moy+15].

Since the ARU-Net was not trained on the original training data, it is hard to compare its results to the other ones. Nevertheless, we would like to stress the fact, that trained systems usually perform better if training set and test set are sampled from the same distribution. E.g., the ARU-Net trained on the cBAD training set achieves an average F-value of 0.9605 for the Bozen test set, which is worse than the F-value of 0.9750 of the system trained solely on the Bozen training set, see Tab. 4.6. This indicates (but does not prove) the superiority of the presented method over the other methods in Tab. 4.8.

### ICDAR2017 Competition on Layout Analysis for Challenging Medieval Manuscripts (Task 2)

The ARU-Net was trained for 250 epochs 1024 samples per epoch on the competition training data<sup>8</sup> provided by the competition organizers, the DIVA-HisDB [Sim+16]. This allows an entirely fair comparison to the participant's results [Sim+17], see

<sup>7</sup>The competition training data was not available to the authors.

<sup>8</sup><http://diuf.unifr.ch/main/hisdoc/diva-hisdb>

Tab. 4.9. The proposed method substantially outperforms the winning one and

Table 4.9.: **Results for the ICDAR2017 Competition on Layout Analysis for Challenging Medieval Manuscripts (Task 2)**— The F-values for Task 2 of all participants and the proposed method are shown for the different subsets of the test set.

Method	CB55	CSG18	CSG863	overall
CVML	0.9534	0.8734	0.9751	0.9340
BYU	0.9597	0.9879	0.9830	0.9768
CITlab	0.9896	0.9853	0.9716	0.9822
proposed	0.9980	0.9828	0.9889	<u>0.9899</u>

reduces the error (the gap to 1.0) by 43.26% (relatively). The specialty of this competition was that the methods should focus on a special kind of text, e.g., comments were not annotated as text. Hence, the ARU-Net had to learn to distinguish between different types of text. The output of the ARU-Net and the detected baselines for a sample image of the CSG18 subset of the test set are shown in Fig. 4.19. One can see, that the ARU-Net entirely ignores all text entities not regarded (in this competition) as main text. Remarkably, no further information besides the input image is provided to the ARU-Net. To demonstrate the capability to adapt to different baseline intuitions, we have trained the ARU-Net on the same images, but all text lines (including comments) were assigned to be text lines. The ARU-Net output for the same image as used in Fig. 4.19 is shown in Fig. A.1. Hence, the ARU-Net yields also impressive results for an entirely different baseline intuition.

### cBAD: ICDAR2017 Competition on Baseline Detection

We compare our average result for the ARU-Net (see Tab. 4.7) to the results presented in [Die+17a], see Tab. 4.10. Our method performs considerably better in both tracks compared to all submissions. Especially, the increase in performance for the complex track is massive. The gap to the maximal F-value of 1 is reduced by nearly 45%. Remarkably, the winning team uses a U-Net based system with task-specific pre- and postprocessing. This indicates that the newly introduced concepts and parametrization, which are presented in this work, significantly improve the capability of the classical U-Net. Some results on chosen images of the cBAD test set are shown in Fig. A.4-A.6. Notably, no further information besides the input image (and the text region information in the simple track) is provided to the ARU-Net nor to the second stage of the workflow during inference.



Figure 4.19.: **Results for an image of the CSG18 subset of the test set** – The original image (only the main text lines were ground truthed), the baseline image generated by the trained ARU-Net and the baselines detected by the proposed method are shown (from left to right).

## 4.5. Conclusion

In this chapter, we have motivated the separation of the text line extraction problem into a baseline detection and a subsequent extraction problem. The baseline detection problem was introduced and a suitable similarity score to evaluate the performance of a baseline extractor was described. Furthermore, we have introduced a baseline detection method which combines modern deep learning techniques and state-of-the-art image processing approaches. The proposed ARU-Net, which is a universal pixel labeling approach, was trained to predict the baseline position and the beginning and end of each text line. This enables the system to handle documents with complex layouts, e.g., tables, marginalia, multi columns layouts. We have shown that the system can be trained from scratch with manageable few training samples for a complex but homogeneous collection. Remarkably, ground truth production is quite cheap. A ground truth sample is just a page with annotated baselines, which can be done in a few minutes per page. Therefore, one can expect that an adaptation on collections, which are not covered by the neural network, is possible with quite reasonable ground truthing effort. Additionally, this is also true for different baseline intuitions. This was demonstrated for two entirely different baseline intuitions for the DIVA-HisDB. Consequently, the proposed baseline extractor has the ability to adapt to different TLIs (regarding the baseline specific questions (a) & (b) of the introduction to this chapter). Hence, it meets the central demand on a BLD as stated in the introduction to this chapter.

The applicability of the proposed method was shown for straight, curved and ori-

Table 4.10.: **Results for the cBAD test set** – The P-, R- and F-values of all participants and of the proposed method for the simple and complex track of the cBAD: ICDAR2017 Competition on Baseline Detection are shown. The bold numbers display the average value for 5 trained ARU-Nets.

Method	Simple Track			Complex Track		
	P-Val	R-Val	F-val	P-Val	R-Val	F-val
LITIS	0.780	0.836	0.807	–	–	–
IRISA <sup>†</sup>	0.883	0.877	0.880	0.692	0.772	0.730
UPVLC	0.937	0.855	0.894	0.833	0.606	0.702
BYU	0.878	0.907	0.892	0.773	0.820	0.796
DMRZ	0.973	0.970	0.971	0.854	0.863	0.859
proposed	<b>0.977</b>	<b>0.980</b>	<b><u>0.978</u></b>	<b>0.926</b>	<b>0.918</b>	<b><u>0.922</u></b>

<sup>†</sup> This method is based on the work presented in [Ren+17].

ented text lines as well as for a combined scenario. The superiority of the proposed ARU-Net in the two-stage workflow over the classical U-Net and over a simplified workflow was shown and statistically verified. Finally, we showed that the proposed method substantially outperforms the previous state of the art. Nevertheless, as one can see in Fig. A.4-A.6 there are still errors made by the system, e.g., missed baselines (see Fig. A.5 – bottom right), segmentation errors (see Fig. A.6 – bottom left), false positives (see Fig. A.4 – top left) or problems with strongly degraded documents (see Fig. A.5 – top left). But these errors do not seem to follow a certain deterministic principle, which is not surprising for a method based on machine learning.

In the next chapter, we will determine a polygonal chain representation of a text line given its baseline. This will be done utilizing the principle of dynamic programming. The entire text line extraction pipeline will then be tested concerning its accuracy at pixel level as well as in conjunction with an ATR system by means of the character error rate.

## 5. Baseline to Text Line

In the literature, different text line representations were utilized for different ATR/KWS systems, e.g., bounding boxes [SF16], x-height areas [Ren+17] or more precise polygonal representations following all ascenders and descenders [Str+16a]. However, Romero et al. [Rom+15] show that the HTR accuracy is not significantly effected by the polygon surrounding the text lines. Even simple strategies to construct surrounding polygons given baseline representations lead to satisfying results.

Here, we will present an approach to calculate polygonal chain representations by utilizing dynamic programming [BD62]; [Cor09] to calculate separating seams which constitute the polygonal chain representation of a text line. Dynamic programming was already utilized in the context of text line extraction, e.g., by Nicolaou et al. [NG09], Saabni et al. [SAE14] or Arvanitopoulos et al. [AS14].

After introducing the seam carving methodology, we will evaluate its performance on a recently hosted competition for text line extraction. Furthermore, we will evaluate the performance in conjunction with a subsequent ATR module. Therefore, several ATR systems were trained on two datasets to investigate the influence of the proposed text line extraction method in an end-to-end fashion.

### 5.1. Seam Carving

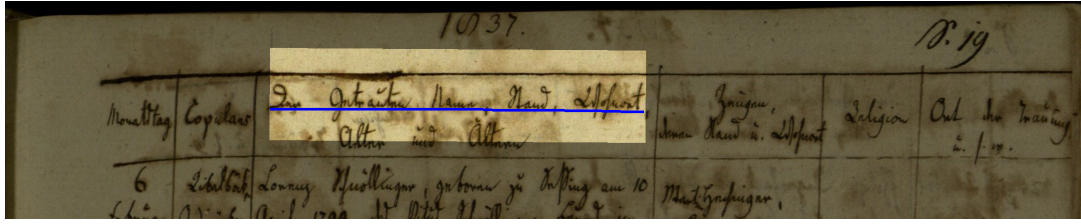
After introducing the term seam, we will briefly describe the proposed seam carving methodology which is basically very similar to the method proposed in [SAE14].

**Definition 5.1.1** (seam). Let  $I \in \mathfrak{I}$  be an image. We call a polygonal chain  $\mathcal{P} = (\mathbf{p}^{(1)}, \dots, \mathbf{p}^{(2I)})$  *seam* iff

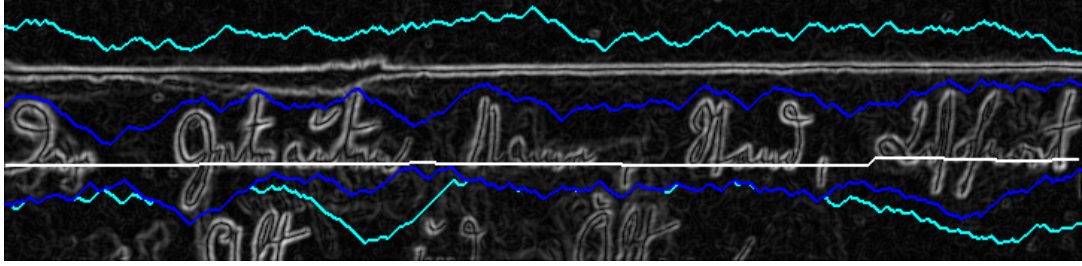
- $\mathbf{p}_2^{(1)} = 1 \wedge \mathbf{p}_2^{(i+1)} = \mathbf{p}_2^{(i)} + 1 \quad \forall i \in [2I - 1],$
- $\mathbf{p}_1^{(i)} \in [1I] \quad \forall i \in [2I] \wedge \left| \mathbf{p}_1^{(i+1)} - \mathbf{p}_1^{(i)} \right| \leq 1 \quad \forall i \in [2I - 1].$

That means, a seam is a polygonal chain which connects the left with the right image border. Furthermore, there has to be a pixel for each column and adjacent pixels are allowed to differ in their  $y$ -coordinates by at most 1.

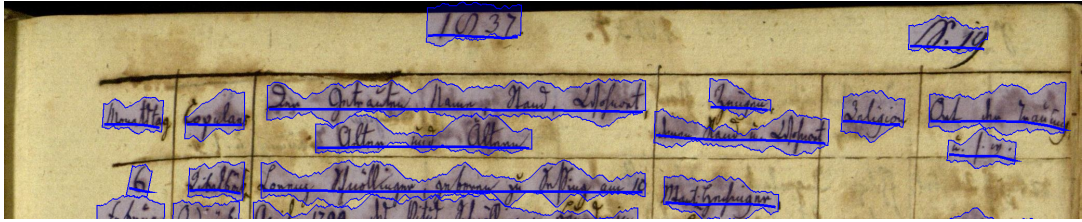
Basically, seam carving aims at the calculation of energy minimal seams. The energy will be defined in a way that a seam which separates two adjacent text lines gets a



(a) Depicted is a baseline  $\mathcal{P}$  (blue) with its corresponding subimage  $I^{(\mathcal{P})}$  (bright area).



(b) The Sobel image with medial seam (white) and separating seams (cyan/blue) which result from a calculation without/with spring model.



(c) Shown are the resulting text lines which were calculated given just the detected baselines.

Figure 5.1.: **Baseline to text line** – This figure illustrates different stages of the proposed baseline to text line methodology.

low energy. Finally, the polygonal chain representation of a text line is built by two seams which separate a text line from its upper and lower neighbors. This approach implies that text lines of interest are roughly horizontal oriented and range from the left image border to the right one, see [NG09]; [SAE14]; [AS14].

Of course, this is not ensured in common scenarios. Therefore, we generate text line specific subimages which satisfy the requirements for the seam carving approach. For this purpose, we utilize the text line's baselines. Let  $\{\mathcal{P}^{(1)}, \dots, \mathcal{P}^{(n)}\} \in 2^{\mathcal{P}_{BL}}$  be the set of baselines in an image which were either automatically detected or manually annotated by an operator. The baseline specific subimages of the original image are calculated utilizing the baseline dependent polygonal chain orientation and tolerance value, cf. Sec. 4.3. The document image is rotated by the polygonal chain orientation. Finally, the text line specific image  $I^{(\mathcal{P})}$  for baseline  $\mathcal{P}$  is the subimage of  $I$  which has a width equal to the baseline width, a height determined by the tolerance value, and is centered at the baseline, see Fig. 5.1a.

To reduce the effect of background noise as well as changes in the illumination, we convert the gray scale image into a Sobel image. To concentrate on changes in the

pixel intensities (edges) which characterize text, the image  $I^{(\mathcal{P})}$  is convolved with the  $x$ - and  $y$ -Sobel filter, respectively. The Sobel filter is an edge detection filter and constitutes the basis for the Sobel image.

**Definition 5.1.2** (Sobel image). Let  $I$  be an image and

$$S^{(x)} = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}, \quad S^{(y)} = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

are the  $x$ - and  $y$ -Sobel filter. The image defined as

$$\text{sob}(I) := \frac{1}{8} \cdot (|S^{(x)} * I| + |S^{(y)} * I|) \in \mathfrak{I}$$

is called *Sobel image*.

An exemplary Sobel image  $\text{sob}(I^{(\mathcal{P})})$  is depicted in Fig. 5.1b. For each Sobel image a *medial seam*  $\mathcal{P}^{(med)}$  which specifies the text line (cf. [SAE14]) is calculated based on the detected baseline, see Fig. 5.1b. Finally, two *separating seams*, one ( $\mathcal{P}^{(s_a)}$ ) above the medial seam and one ( $\mathcal{P}^{(s_b)}$ ) beneath it, are calculated utilizing dynamic programming. The cost function  $C$  to be minimized is determined by the pixel intensities in the Sobel image and by the distance to the medial seam. For a Sobel image  $I$ , a medial seam  $\mathcal{P}^{(med)} = (\mathbf{q}^{(1)}, \dots, \mathbf{q}^{(2I)})$  and an arbitrary seam  $\mathcal{P} = (\mathbf{p}^{(1)}, \dots, \mathbf{p}^{(2I)})$ ,  $C$  is defined as

$$C(\mathcal{P}; I, \mathcal{P}^{(med)}) := \sum_{i=1}^{2I} I_{\mathbf{p}^{(i)}} + k \cdot |\mathbf{p}_y^{(i)} - \mathbf{q}_y^{(i)}|.$$

The second term ensures that the medial seam attracts the separating seam. It is referred to as *spring model* [SAE14]. For a high spring factor  $k$  the separating seams tend to include solely text line components, but also tend to cut off superscript dots. On the other hand, a low spring factor results in separating seams which include background noise and other elements which are not of interest. In Fig. 5.1b the separating seams for a spring factor  $k = 0$  (cyan) and  $k = 0.005$  (blue) are shown. The influence of the spring factor is further investigated in the next section.

## 5.2. Baseline to Text Line Experiments

In this section, we evaluate the introduced method on Task 3 of the ICDAR2017 Competition on Layout Analysis for Challenging Medieval Manuscripts [Sim+17]. Task 2 of this competitions was already mentioned in Sec. 4.4. In Task 3, the competitors had to submit methods which produce polygonal representations of the text lines given the input images. Because the text lines were annotated at



pixel level, see [Sim+16], an evaluation at pixel level is possible. Consequently, the methods were evaluated by means of the *intersection over union* methodology (*IU*) [MB01]. The IU is defined as

$$IU := \frac{TP}{TP + FP + FN}$$

with the true positives (*TP*), the false positives (*FP*), and the false negatives (*FN*). Hence, the fraction of the correct detections (intersection of all detections and the ground truth) and all detections plus the missed ones (union over all detections and the ground truth) is calculated. A detailed description of the resulting evaluation scheme can be found in [Alb+17]; [Sim+17]. Here, we just point out some essential parts. The evaluation tool is available as standalone jar<sup>1</sup>.

It has to be mentioned that only foreground pixels (which were assigned to be foreground by some operator) inside the hypothesis and ground truth polygonal chains were taken into account for the *IU* calculations. Finally, the methods were evaluated with respect to three different scores, namely the pixel *IU*, the matched pixel *IU*, and the line *IU*. The latter constitutes the main score which determines the final ranking for the competition. Consequently, the line *IU* constitutes the similarity score as introduced in Sec. 1.3. For the line *IU*, *TP* is the number of correctly detected lines, *FP* is the number of extra lines, and *FN* is the number of missed lines. A line is said to be correct if there is a pair of GT and HYP line which has a pixel precision and a pixel recall value greater than some threshold (0.75 in the competition). If the precision is lower than this threshold, the line is said to be a false positive. In the last case, for a recall which is lower than the threshold the line is a false negative. Pixel *IU* and matched pixel *IU* indicate how well the lines were extracted at pixel level, instead as on line level. For a detailed description, we refer to [Sim+17]. In this work, we will just provide the line *IU* (*LIU*) and the pixel *IU* (*PIU*) of all methods.

In the following, we demonstrate the capability of the introduced method to extract text lines in complex scenarios in a high quality. On the other hand, we also show that the performance strongly relies on the spring factor. We extract the text lines as described in the previous section for baselines detected by the method described in Chap. 4 (the baseline detection was evaluated in Tab. 4.9).

As mentioned in [Sim+17]: *The presence of interlinear glosses is usually a major difficulty for obtaining a good text line segmentation.* This is especially true for some images of the sets CSG18 and CSG863. The ARU-Net was trained to detect main body text only. As a result, the interlinear glosses cause errors in the seam carving text extraction. Interlinear glosses which are close to the main body text lines tend to be assigned to the main body text lines for low spring factors. It is plausible that these errors diminish for increasing spring factors. The seams are forced to be close to the correctly detected baselines. Consequently, the interlinear glosses are

<sup>1</sup>[https://github.com/DIVA-DIA/DIVA\\_Line\\_Segmentation\\_Evaluator](https://github.com/DIVA-DIA/DIVA_Line_Segmentation_Evaluator)

not assigned to the main body text lines any more. On the other hand, correct pixels are also cut from the main body text lines.

This behavior is encoded in the values in Tab. 5.1. For the CB55 set which does not comprise any interlinear glosses, the *LIU* value as well as the *PIU* value are quite good for small spring factors. There are no interlinear glosses which could be erroneously assigned to the main body text lines. But for increasing spring factors more and more correct pixels are cut (*PIU* gets smaller) until the precision/recall values fall below the threshold and the *LIU* also decreases. For the other two subsets, the behavior is different. *LIU* as well as *PIU* are low for small spring factors (many interlinear glosses are erroneously assigned to the main body text lines). This problem reduces with increasing factors up to a certain point where too many correct pixels were cut from the text lines. The entire behavior is visualized in Fig. A.7 - A.9.

Table 5.1.: **Results for the ICDAR2017 Competition on Layout Analysis for Challenging Medieval Manuscripts (Task 3)** – The line *IU* and pixel *IU* of all submissions, of an additional combined method, and of the proposed method for several spring factors are shown for the three subsets of the test set. Finally, the average value is given. The average line *IU* constitutes the basis for the competition ranking.

Method	CB55		CSG18		CSG863		average	
	<i>LIU</i>	<i>PIU</i>	<i>LIU</i>	<i>PIU</i>	<i>LIU</i>	<i>PIU</i>	<i>LIU</i>	<i>PIU</i>
IAIS	5.67	30.53	39.17	54.52	25.96	46.09	23.6	43.71
BYU	84.29	80.23	69.57	75.31	90.64	93.68	81.5	83.07
CITlab	99.33	93.75	94.90	94.47	96.75	90.81	96.99	93.01
Combined <sup>†</sup>	98.04	96.67	96.91	96.93	98.62	97.54	97.86	97.05
$k = 0.0$	99.68	97.17	59.58	78.03	85.95	90.12	81.74	88.44
$k = 0.001$	99.68	96.79	82.61	88.22	87.56	90.67	89.95	91.89
$k = 0.005$	99.68	96.22	94.92	94.24	95.73	93.28	96.78	94.58
$k = 0.01$	99.68	95.3	96.69	95.41	96.38	94.22	97.58	94.98
$k = 0.02$	99.35	92.62	96.97	95.57	96.70	94.67	97.67	94.29
$k = 0.05$	80.17	80.60	97.29	93.78	96.74	90.95	91.4	88.44
$k = 0.001\ddagger$	99.68	96.62	97.29	97.01	97.07	95.56	<u>98.01</u>	96.40

<sup>†</sup> This method was proposed by the organizers, who combined a Task 1 method with an own text line extraction method. It was noncompetitive.

<sup>‡</sup> The baselines of the interlinear glosses which were detected by the ARU-Net trained to detect all text lines (Sec. 4.4.4 & Fig. A.1) were taken into account for seam carving.

However, it is much more meaningful to directly incorporate the knowledge about the interlinear glosses. Note that this was also done in the "Combined" method of the competition organizers, see Tab. 5.1. Here, we propose a slightly different approach. Instead of labeling all pixels, we make use of the ARU-Net which was trained to detect all text lines, see Sec. 4.4.4. After extracting all text lines, we eliminate the text lines which were not detected by the ARU-Net trained to detect main body text lines only. The resulting system yields the best results concerning the line *IU* (similarity score). Some representative results are shown in Fig. 5.2. Consequently, the competitiveness of the proposed approach was demonstrated. In the following, we fix the spring factor to be 0.001.

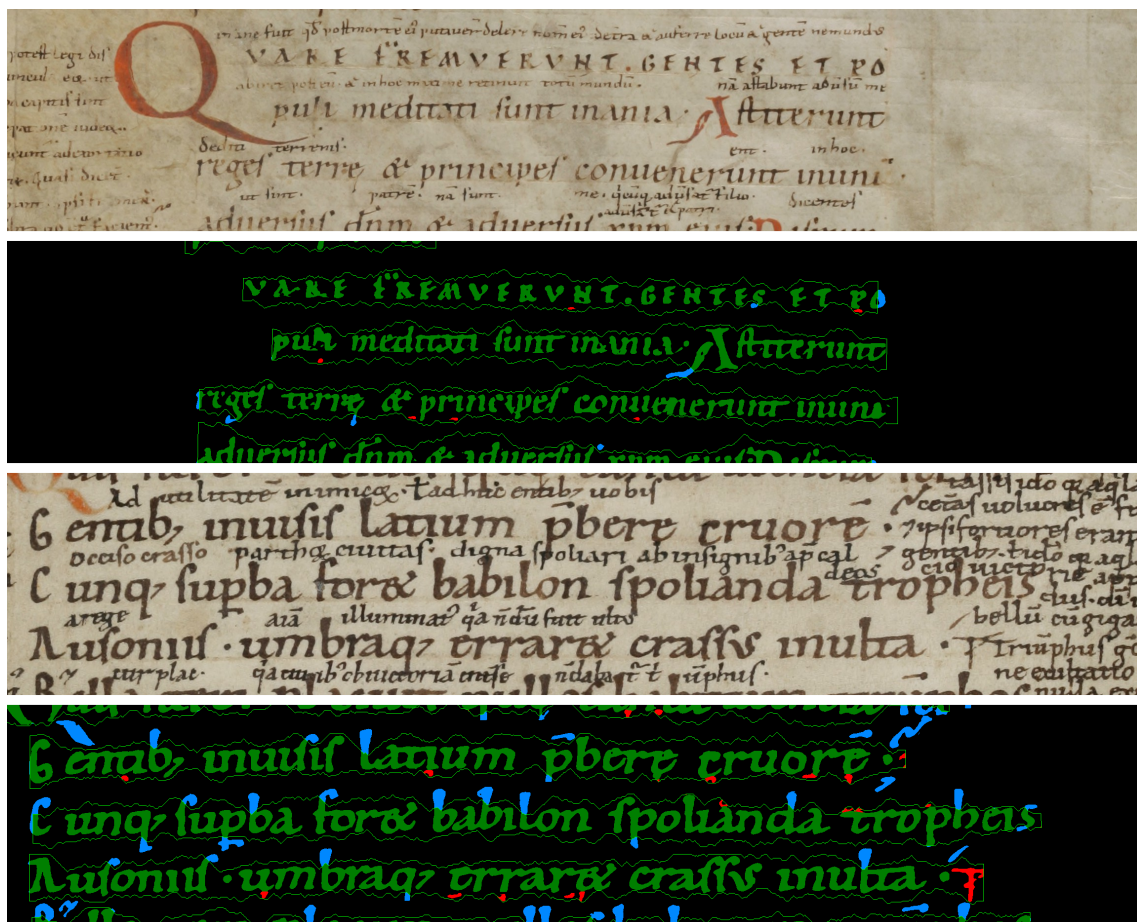


Figure 5.2.: Text line extraction results for two snippets of the DIVA-HisDB – Shown are the results of the text line extraction which takes care of the interlinear glosses (see main text) for two snippets of the DIVA-HisDB test set [Sim+16]. A spring factor of 0.001 was used. The visualization was generated by the tool presented in [Alb+17]. Green means correct, blue are false negatives, yellow pixels are assigned to the wrong text line and red pixels are no main body pixels at all.

### 5.3. End-to-end Experiments

After demonstrating that the proposed approach is at least comparable to the state of the art concerning a pixel level based similarity score, we will investigate the entire text line extraction pipeline in an end-to-end fashion. Therefore, we have trained a hybrid conv-BLSTM ATR system and evaluated the entire information retrieval pipeline for two different data sets.

Therefore, the text lines are extracted in the way described in the last chapters. Afterwards, the extracted text line images were preprocessed in the way described in [Str+16a] before fed into the ATR system. Hence, they have a fixed height of 64px. We will not describe the conv-BLSTM ANN in detail. For a comprehensive description of such systems see [Pui17]; [SBY17]; [Bre17]; [BM17]. Here, we will just outline the differences of our system. Basically, our system consists of 3 conv layers (Kernel Dim-Stride Dim-Pooling Dim:  $4 \times 2 \times 1 \times 8 - 4 \times 2 - 1 \times 1$ ;  $4 \times 2 \times 8 \times 32 - 1 \times 1 - 4 \times 2$ ;  $3 \times 3 \times 32 \times 64 - 1 \times 1 - 1 \times 2$ ), followed by 3 blstm layers (256 units each), followed by a fully connected softmax layer. Batch normalization (see [IS15]) and local response normalization (see [KSH12]) are applied after each convolutional layer. The BLSTM layers are trained with dropout 0.5 applied to the output of each cell (see [GG16]). Finally, we have trained the networks for a fixed epoch size (150 with 8192 samples per epoch) utilizing an exponential moving average for the trainable model parameters. To increase the amount of training data, we employed some data augmentation strategies (grid deformations [Wig+17], dilation, and erosion) for the preprocessed input images.

The experiments were performed for two different datasets. The Bozen dataset which was already described in Sec. 4.4 and the StAZH dataset. The StAZH dataset is composed of samples chosen from 800 pages of the Staatsarchiv des Kantons Zürich (StAZH) collection. This set consists of Kantonsratsprotokolle and Regierungsratsbeschlüsse whereas the former contains less than 10% of the overall amount of text. The documents were written in German current in a time between 1803 and 1882. The texts consist of resolution/enactment of the cabinet as well as the parliament of the state of Zurich (starting in 1848 “canton”). 320 pages were chosen for training, 40 for testing.

In case of the StAZH dataset, ground truth is available at baseline level. Hence, we applied the presented baseline to text line algorithm on the entire dataset to generate feasible training data for the ATR system. We trained our ATR system 5 times on the generated training data. Furthermore, we trained our baseline detection system 5 times on the training data to detect the baselines. Finally, the 5 trained ATR systems were applied on the extracted text lines of the test set which were generated given the ground truth baselines (GT-BL). Afterwards, the 5 trained ATR systems were applied on the extracted text lines of the test set which were generated given just the raw input document images (no GT). For each ATR system, a different baseline detection system was used in the no GT scenario. We evaluated the quality

by means of the average character error rate and the average bag of words  $F_1$ -score over the 5 experiments. Remarkably, the transcription was calculated given the raw ANN output, no dictionary or language model was utilized.

Table 5.2.: **Results for the end-to-end experiments** – Shown are the average character error rate and the average bag of words  $F_1$ -score for several experiments. GT-TL means that the ATR system was evaluated on the GT text lines, GT-BL means an evaluation on the text lines extracted given the GT baselines, and no GT means that the text lines were extracted given solely the input image. For the Bozen dataset, the results of [San+16] are listed to provide some comparison (comparable to GT-TL).

Dataset	System	Evaluation	
		CER in % [min,max]	$F_1$ -score (BoW) [min,max]
StAZH	GT-TL	<b>3.27</b> [3.23, 3.32]	<b>0.890</b> [0.886, 0.891]
	no GT	<b>3.97</b> [3.86, 4.38]	<b>0.891</b> [0.890, 0.893]
Bozen	ParisTech	<b>18.5</b> $\pm$ 0.5 <sup>‡</sup>	–
	LITIS	<b>7.3</b> $\pm$ 0.4 <sup>‡</sup>	–
	A2IA <sup>†</sup>	<b>5.4</b> $\pm$ 0.3 <sup>‡</sup>	–
	BYU	<b>5.1</b> $\pm$ 0.3 <sup>‡</sup>	–
	RWTH <sup>†</sup>	<b>4.8</b> $\pm$ 0.3 <sup>‡</sup>	–
	GT-TL	<b>5.11</b> [4.94, 5.20]	<b>0.850</b> [0.843, 0.858]
	GT-BL	<b>5.31</b> [5.18, 5.39]	<b>0.841</b> [0.835, 0.846]
	no GT	<b>5.97</b> [5.84, 6.16]	<b>0.831</b> [0.828, 0.834]

<sup>†</sup> A language model was used on top of the ATR model.

<sup>‡</sup> It is not clarified in [San+16] what is meant by  $\pm x$ .

For the Bozen dataset, baselines **and** polygonal chain representations of the text lines are available as ground truth. In addition to the above mentioned experiments, we trained 5 ATR systems on these ground truth text lines (GT-TL). Since the Bozen dataset was used for an ATR competition [San+16], we compare our results to the participant's results. Note that in this competition just the ATR performance was evaluated. The ground truth text lines were provided to all participants by the organizers. Consequently, a comparison to our GT-TL scenario is fair.

Note that for the CER calculation in the no GT scenario the alignment algorithm of Sec. 4.2 (Alg. 5) was utilized. It was slightly adapted such that the edit distance between GT and HYP is minimized. Consequently, segmentation errors are penalized twice, i.e., if a text line is falsely segmented, one of its parts is aligned to the GT which results in insertions, the other part is not aligned at all which results in deletions. The results of the experiments are summarized in Tab. 5.2.

The results demonstrate the usability of the proposed text line extraction method for a fully automated information retrieval pipeline. Of course, if the task gets complex (no GT scenario) the error increases. Nevertheless, the error is dominated by the ATR error. The text line extraction is responsible for a relative increase in the CER of 21.5% (StAZH) and 16.8% (Bozen), respectively. For the StAZH experiments, a further analysis of the results allows for an insight in the type of errors. The increase in the CER is larger than for the Bozen dataset. Nevertheless, for the bag of words  $F_1$ -score it is vice versa. This indicates that the errors in the StAZH experiments are due to segmentation errors. Indeed, sometimes the marginalia is connected to the main body text. This results in an increase of the CER but all words are correctly transcribed. Hence, the bag of words  $F_1$ -score does not decrease. This phenomena is shown for an exemplary page in Fig. A.10-A.11. The errors for the Bozen dataset are not as easy to analyze. The scenario is much more complex and the errors seem to be the result of several issues, e.g., ATR model noise, segmentation issues, detection errors, cutted ascenders/descenders. However, the results for an exemplary page are shown in Fig. A.12-A.13. Remarkably, comparable ATR results are achieved with entirely different polygonal chain representations (TLIs). Note that it is of importance that the TLI used for training is also used for testing. For instance, an ATR system trained on the TLI visualized in Fig. A.12 and tested for the TLI visualized in Fig. A.13 yields a CER of 7.04% (compared to 5.96% for an ATR system trained on the TLI visualized in Fig. A.13).

## 5.4. Conclusion

In this chapter, we have introduced a fully automatic baseline to text line method which is based on the principle of dynamic programming. Given the text line's baselines it automatically calculates polygonal chain representations. These representations can be used to train an ATR system. We evaluated this method as a stand alone method at pixel level and in an end-to-end fashion by means of CER and bag of words  $F_1$ -score for a subsequent ATR system. In both cases, the proposed method demonstrated satisfying results. Of course, it is responsible for a slight increase in the CER compared to an ATR system which works on the GT text lines. Nevertheless, the CER is dominated by the errors made by the ATR system.



## 6. Conclusion & Outlook

There is no doubt that the general public directly benefits from an automation of the information retrieval pipeline in the context of scanned historical documents. This dissertation contributes a small piece to achieve the long-term goal to automatically access the cultural heritage which is hidden in the billions over billions of scanned document images which are stored on servers all over the world. This thesis formulates the text line extraction problem for arbitrary document images in a formal way and describes a sophisticated text line extraction method which constitutes the current state of the art. The method is already usable and used by a wide audience of researchers via the Transkribus<sup>1</sup> platform.

The developed method is based on a subsequent solution of the problems of baseline detection and text line extraction given the detected baselines. The baseline detection methodology constitutes the main part of the thesis. The developed baseline detection method is a combination of state-of-the-art machine learning approaches and classical image processing techniques. The method and the basics of the used concepts are thoroughly motivated and introduced in a quite formal way. The baseline detector demonstrated its applicability for various different datasets and difficulties in an impressive quality, cf. Sec. 4.4. Because there was no suitable similarity score to evaluate and compare the performance of baseline detection methods, this thesis motivates and introduces a similarity score to quantify the similarity between two sets of baselines. The developed score was already used to rank the submissions of two recently hosted international competitions on baseline detection.

The property of the baseline detector to be trainable given appropriate training samples is inherited from the machine learning component and allows not only for an adaptation to different collections but also for an adaptation to different understandings about what a text line is and how text lines should look like. It is demonstrated that this adaptation is possible without any expert knowledge just by retraining, see Sec. 4.4.4. Basically, this is the major advantage of the developed approach and allows for a usage at a large scale by a wide audience.

Finally, a text line extraction method based on the detected baselines is introduced. This method is based on the well-known seam carving approach which itself is based on dynamic programming. The entire text line extraction pipeline is evaluated twice. First, it is evaluated on pixel level. Second, it is evaluated in an end-to-end fashion. For this purpose, an state-of-the-art ATR system was trained to transcribe the

---

<sup>1</sup><https://transkribus.eu>



content of the extracted text lines. It was shown that the overall character error rate is dominated by the ATR errors and not by errors due to extraction issues.

Although the developed approach outperforms the previous state of the art and that the transcription errors are dominated by ATR errors instead of segmentation errors, there is still scope for improvement. It is stated in Sec. 1.3 that we aim for a design of an optimal text line extractor which is adaptable to different text line intuitions without any expert knowledge. Finally, we can state that the designed method is adaptable but not optimal since, e.g., the F-value in Sec. 4.4 does not reach the maximum value of 1. Consequently, there are still errors made by the system as already discussed in Sec. 4.5. However, after comparing the developed method to various other methods on several different datasets using two similarity scores, we conclude the proposed method is maybe not optimal but it outperforms (to date) all its competitors.

In future work, we will investigate several new concepts, e.g., capsules to improve the performance for rotated text lines, inception blocks to increase the representative power of the model, deeply supervised networks to accelerate the training, ... . However, we will also focus on the redesign of the baseline to text line methodology. As shown in Sec. 5.3 even a quite simple polygonal chain representation is sufficient to yield impressive ATR results (which could be better than the results achieved with the proposed text line representation). Hence, the investigation of the interaction of text line extractor and ATR/KWS system is of special importance to improve the overall performance of the information retrieval pipeline and will be of special importance for future research.

# A. Appendix

## A.1. Exemplary Results

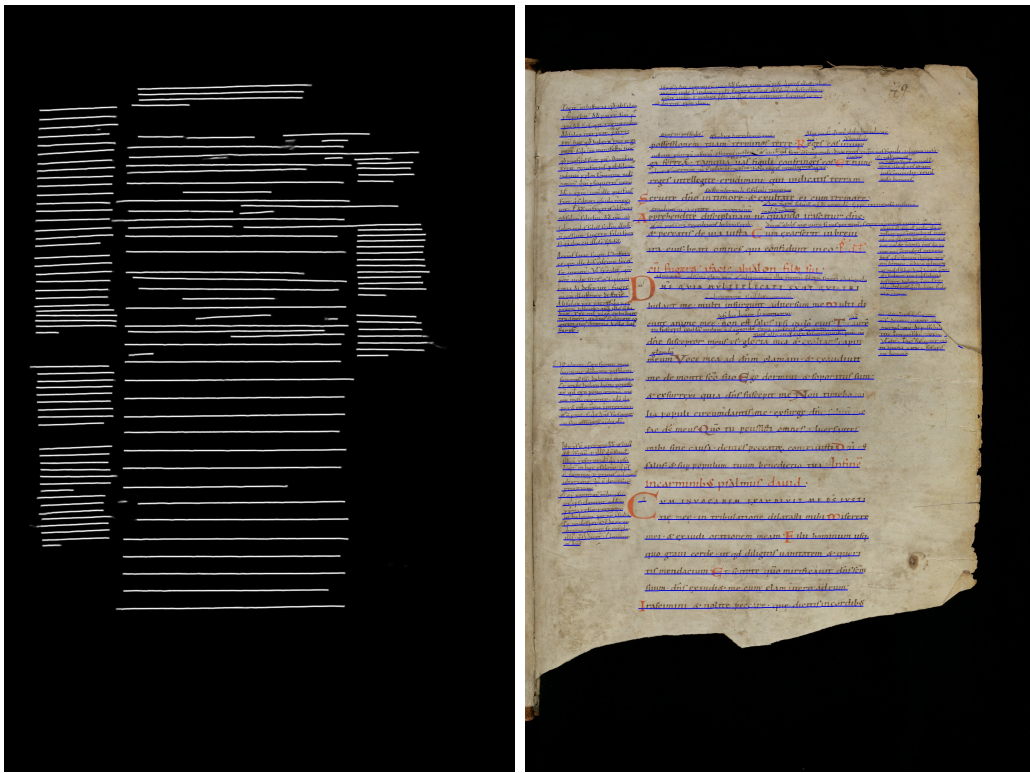


Figure A.1.: Results for an image of the CSG18 subset of the test set with an alternative BLI – Shown is the baseline image generated by the trained ARU-Net and the baselines detected by the proposed method for the image used in Fig. 4.19. In contrast, here, the ARU-Net was trained to detect **all** text lines, not just the main body text lines.

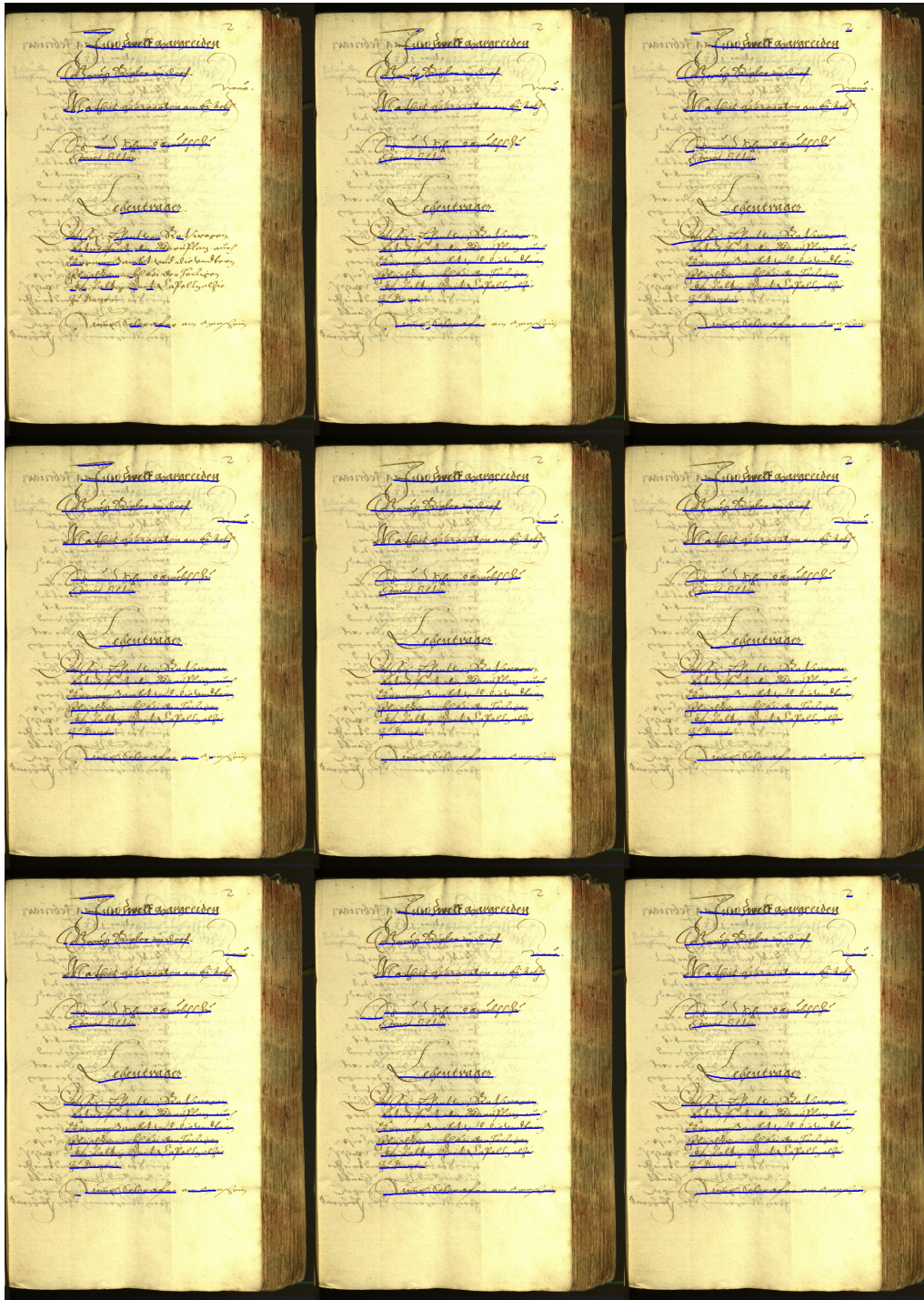


Figure A.2.: **Results for an image of the Bozen test set** – Results for RU-Nets trained on 5, 30 and 350 training samples (left to right) with different data augmentation strategies B, S+A and S+A+E (top to bottom) are shown.





Figure A.3.: **Results for an image of the Bozen test set** – Results for two “degraded” images are shown. The images were arbitrarily curved and rotated.











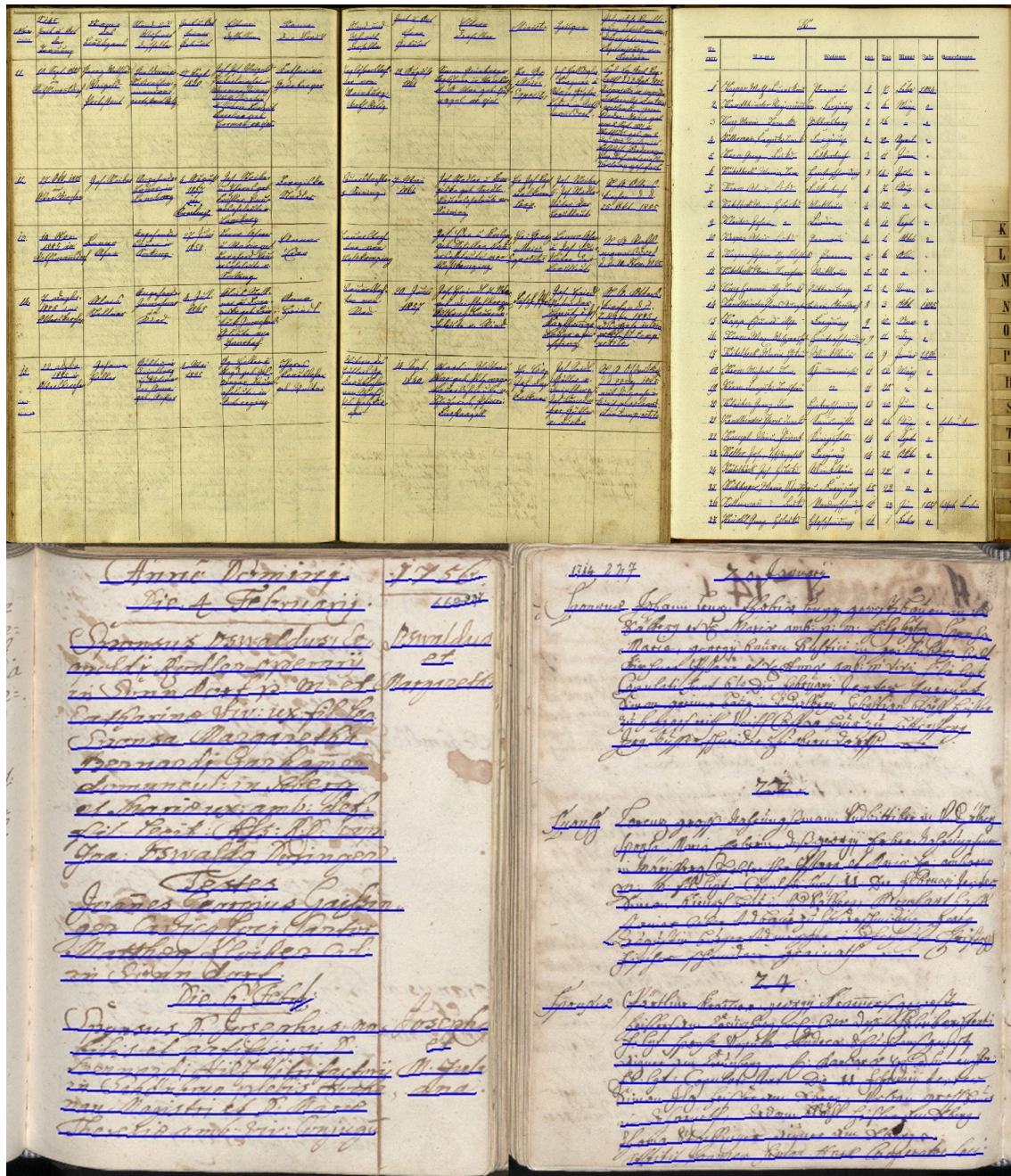


Figure A.6.: Results for images of the cBAD test set [Die+17b] – The images are sampled from the complex track (no layout information was given).

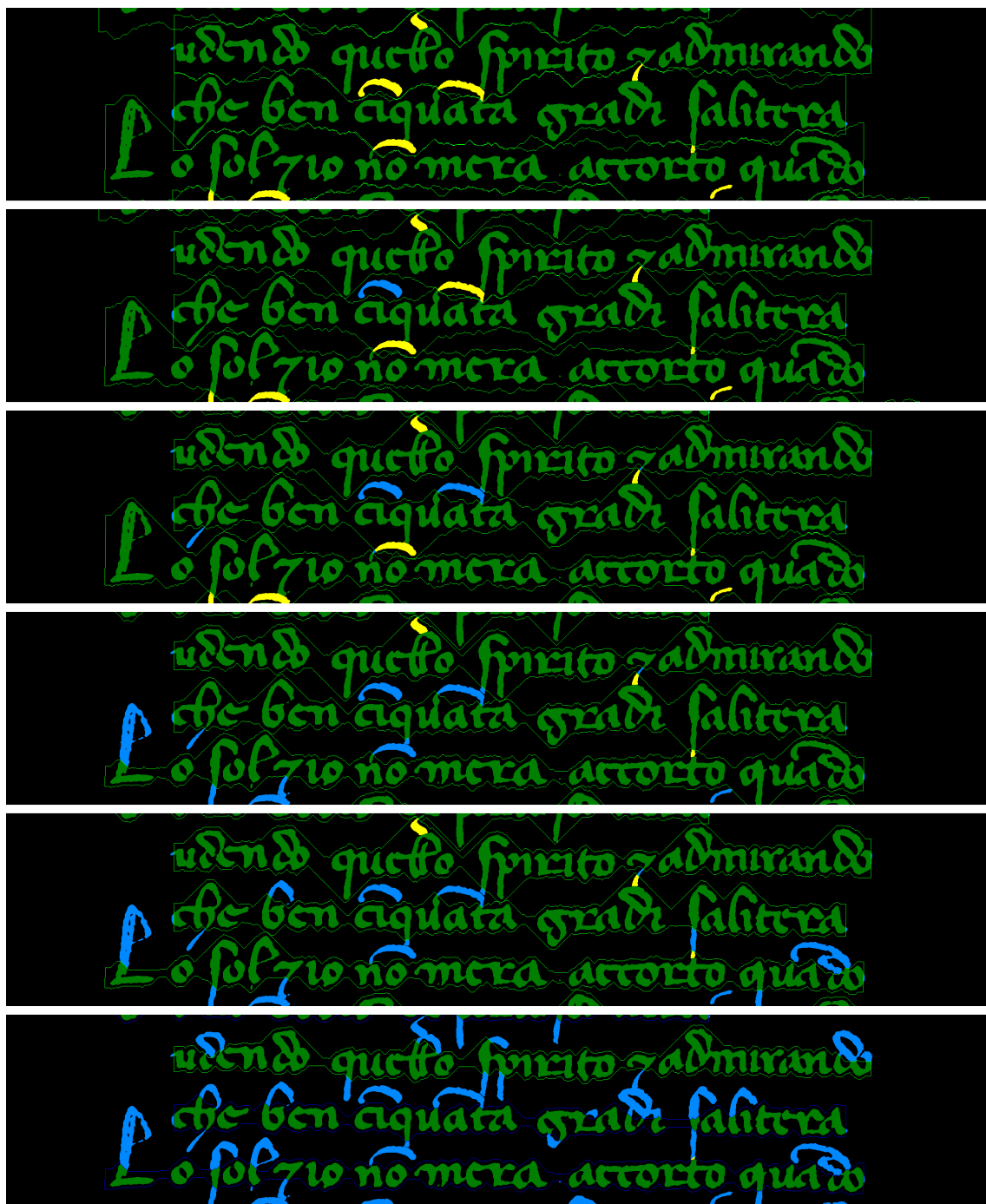


Figure A.7.: Text line extraction results for the CB55 subset – Shown are the results for the text line extraction for a snippet of the CB55 subset of the DIVA-HisDB [Sim+16]. Distance penalty values of (from top to bottom): 0, 0.001, 0.005, 0.1, 0.2, 0.5 were used. The visualization was generated by the evaluation tool presented in [Alb+17]. Green means correct, blue are false negatives, yellow pixels are assigned to the wrong text line and red pixels are no main body pixels at all.



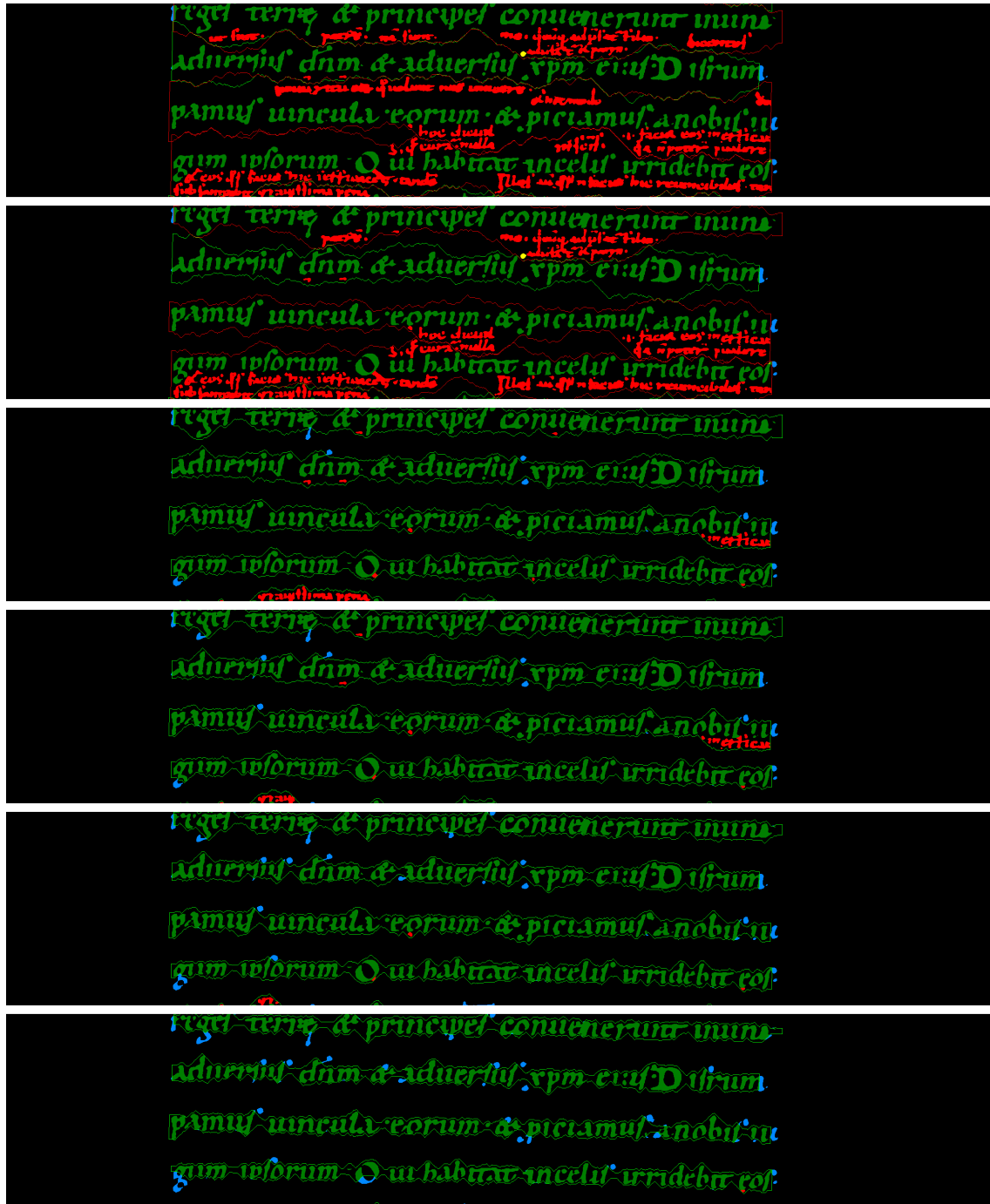


Figure A.8.: Text line extraction results for the CSG18 subset – Shown are the results for the text line extraction for a snippet of the CB55 subset of the DIVA-HisDB [Sim+16]. Distance penalty values of (from top to bottom): 0, 0.001, 0.005, 0.1, 0.2, 0.5 were used. The visualization was generated by the evaluation tool presented in [Alb+17]. Green means correct, blue are false negatives, yellow pixels are assigned to the wrong text line and red pixels are no main body pixels at all.

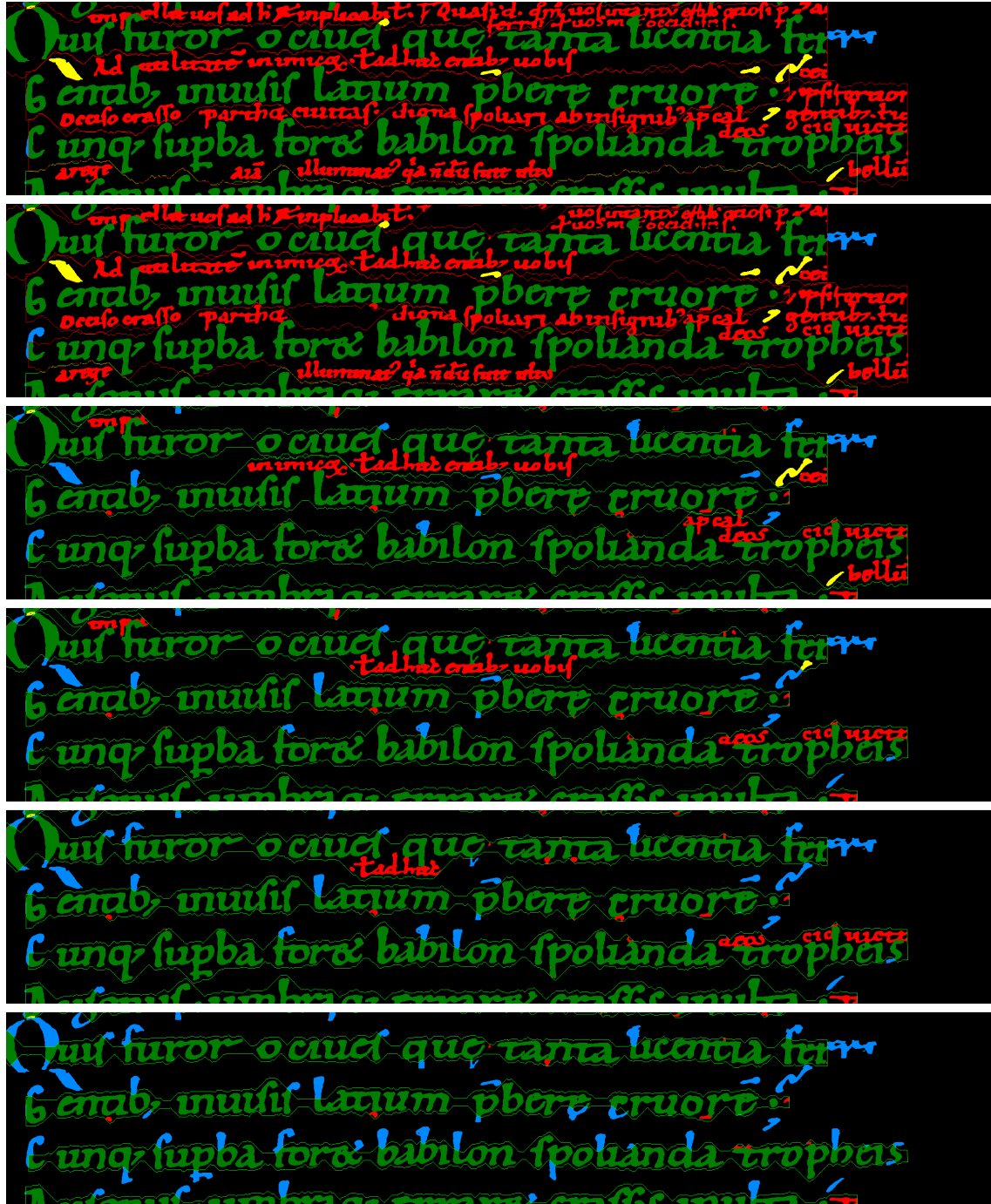


Figure A.9.: Text line extraction results for the CSG18 subset – Shown are the results for the text line extraction for a snippet of the CB55 subset of the DIVA-HisDB [Sim+16]. Distance penalty values of (from top to bottom): 0, 0.001, 0.005, 0.1, 0.2, 0.5 were used. The visualization was generated by the evaluation tool presented in [Alb+17]. Green means correct, blue are false negatives, yellow pixels are assigned to the wrong text line and red pixels are no main body pixels at all.

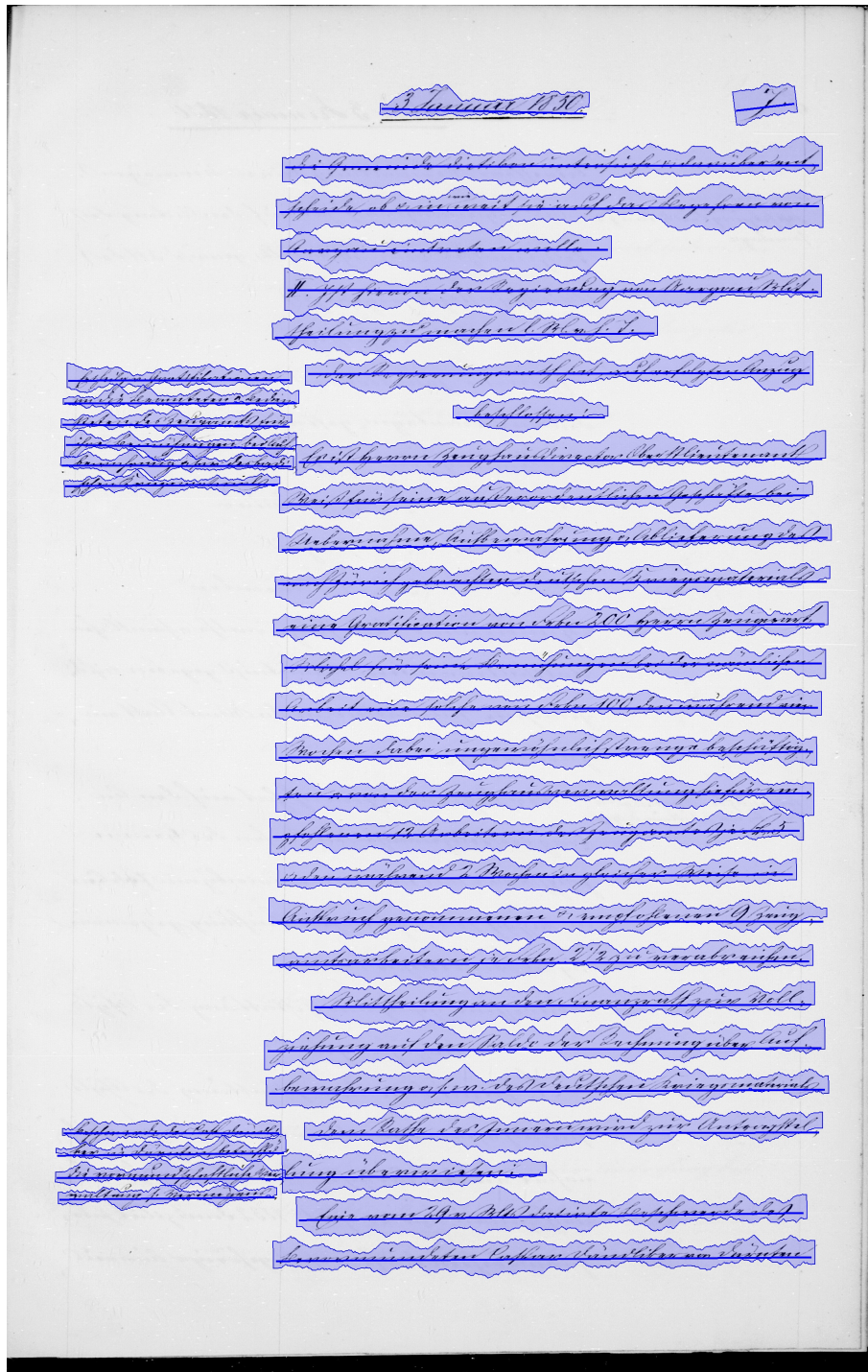


Figure A.10.: Text line extraction result (GT-BL) for the StAZH dataset  
 – Shown are the text line extraction results for an image of the StAZH dataset given the ground truth baselines. The trained ATR system yields an CER of 5.80% for the depicted text lines.



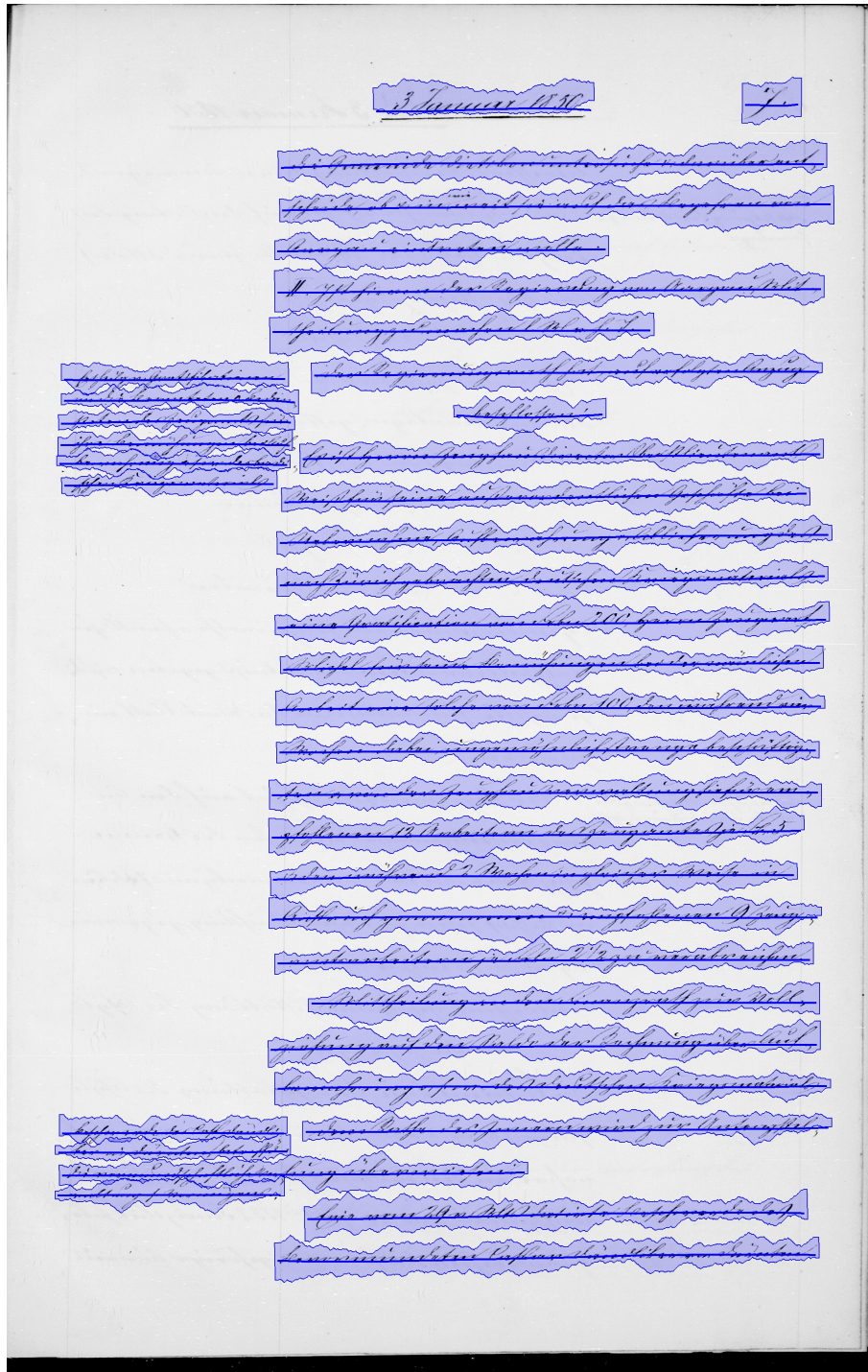


Figure A.11.: **Text line extraction result (no GT) for the StAZH dataset** – Shown are the text line extraction results for an image of the StAZH dataset given solely the raw image. The trained ATR system yields an CER of 7.83% for the depicted text lines. The increase in the CER is (mainly) due to the segmentation error connecting marginalia and main body text.









# List of Figures

1.1. <b>Baselines and text lines for an image snippet of [Die+17b]</b> – Illustration of the detected baselines (solid blue lines) and the extracted text lines (blue underlay) for a snippet of a historical document image which were estimated by the proposed text line extraction method. . . . .	3
3.1. <b><math>\alpha</math>-<math>\beta</math>-swap optimization via graph cut</b> – Shown is the optimal $\alpha$ - $\beta$ -swap estimation utilizing the minimum cut. In the initial labeling, $\mathbb{A}^{(\alpha)} \cup \mathbb{A}^{(\beta)} = \{a, b, c, d, e, f, g\}$ are labeled with $\alpha$ (blue) or $\beta$ (cyan). The set of elements which is labeled with $[l] \setminus \{\alpha, \beta\}$ is shown in magenta. The neighborhood system $\mathfrak{N}$ is represented by the black lines. The additional edges which connect the terminals to $\mathbb{A}^{(\alpha)} \cup \mathbb{A}^{(\beta)}$ are shown in gray. The edges leaving the magenta set are solely of interest for the weight function, cf. Eq. (3.2.3), but they are <b>not</b> part of the graph for which the minimum cut is estimated. . . . .	35
4.1. <b>Differences in the text line intuitions</b> – This figure demonstrates the major issues in which TLIs typically differ. . . . .	40
4.2. <b>Polygonal chain normalization</b> – Depicted is the normalization of two different representatives of the same baseline. Only two points of the lower left baseline have a correspondence in the one above. The two intermediate points do not contribute any additional information (straight line). Hence, the normalizations result in two equivalent polygonal chains. . . . .	44
4.3. <b>Off-text distance and in-text distance for two pixels</b> – The concept of off-text distance and in-text distance for two pixels $\mathbf{p}, \mathbf{q}$ and an orientation vector (blue arrow) is displayed. The orientation vector encodes the text direction. The off-text distance is the component of $(\mathbf{p} - \mathbf{q})$ which is orthogonal to the text direction. The in-text distance is the component which has the text orientation. The magenta parallelogram is utilized for an efficient calculation of the distances, see main text. . . . .	45
4.4. <b>Baseline similarity score</b> – Depicted is a snippet of an example document image sampled from [Die+17b] along with GT and HYP baselines as well as intermediate steps of the evaluation scheme. . . .	48



- 4.5. **Effect of the normalization on the similarity score** – Shown are two different scenarios for the similarity score calculation for different distance parameters in the polygonal chain normalization. GT baselines are red, HYP baselines blue. The red circles (radius of  $3t$  with a tolerance value  $t$ , cf. Alg. 4) are the areas of interest for the coverage function. For  $d = 92$  the similarity score would be 0, because the HYP pixels are all outside of the areas of interest. On the contrary, the result for  $d = 46$  would be reasonable. . . . . 54
- 4.6. **Two-stage workflow to detect baselines** – The first stage utilizes a deep artificial neural network to perform a pixel labeling. The result of Stage I is the input for an image processing based method in Stage II. This method clusters superpixels to build baselines. The image is sampled from the cBad complex test set [Grü+18a]. . . . . 56
- 4.7. **U-Net** – The input is an image of arbitrary spatial dimension. "Act" is the activation function thus the rectangles represent sets of activation maps. Each rectangle represents a 3-dim array ( $\in \mathbb{R}^{***}$ ) of certain dimensionality. Within each scale space (roman numbers) the feature map widths and heights are constant (encoded by the height of the rectangles). The number of feature maps  $z$  is pictured by the width of the rectangles. Between adjacent scale spaces the spatial dimension decreases by a certain factor (2 in the figure) and the representative depth (number of feature maps) increases by the same factor. . . . . 58
- 4.8. **Residual Block** – The input is convolved and the resulting 3-dim array (the maps before passed through an activation function are referred to as logits) is used twice. At the first branch it is passed through the activation function and further processed by several convolution layers. At the second branch it is directly fed into a summation node. After a point-wise summation of the two logit maps an activation function is applied. The shortcut enables for an easy identity propagation and error backpropagation. Arbitrarily many inner layers are possible. . . . . 59
- 4.9. **ARU-Net** – The input image and its downscaled versions are fed into the A-Net and R-U-Net (weight sharing accross different scales). The results for the lower resolutions are deconvolved. The attention maps are passed through a softmax normalization. The brighter the map at a certain position the more attention is paid to that position at the corresponding scale. The attention maps are point-wise multiplied with the feature maps of the RU-Net. The results are summed and a classification is performed. . . . . 61
- 4.10. **Baseline and pixel ground truth** – These are shown for the top snippet of the image of Fig. 4.6. . . . . 63
- 4.11. **Superpixel calculation** – The intermediate steps of the superpixel calculation are shown for the top snippet of the image of Fig. 4.6. . . 64

- 4.12. **Local text orientation and interline distance for an SP** – Depicted are the local text orientation and the interline distance for SP  $\mathbf{p}$ . Obviously,  $\mathbf{p}$  belongs to the lower text line. Consequently its local text orientation ( $\omega_{\mathbf{p}}$ ) is roughly  $10^\circ$ . Its interline distance ( $s_{\mathbf{p}}$ ) is the shortest off-text distance from its textline’s baseline to the adjacent text line. . . . . 66
- 4.13. **Delaunay neighborhood system** – Neighborhood system  $\mathfrak{N}$  calculated by Delaunay’s triangulation for the set  $\mathbb{S}$  of SPs of Fig. 4.11d. 66
- 4.14. **Interline distance estimation** – Illustration of several projection profiles for a certain SP (red point). The profiles for different diameters  $d \in \{64, 128, 256, 512\}$  and an orientation of  $0^\circ$  are shown in green. The winning period (interline distance) is drawn as yellow curve. In blue a histogram for a wrong orientation ( $45^\circ$ ) is shown. . . 67
- 4.15. **SPs along with their assigned states** – The local text orientation of each SP is visualized by the orientation of the green lines (for better clarity rotated by  $90^\circ$ ). The length of the lines encode the interline distance of the corresponding SP. . . . . 70
- 4.16. **Regression curve and cluster distance** – Displayed are two clusters  $\mathbb{S}^{(1)}$ ,  $\mathbb{S}^{(2)}$  as blue points. Their corresponding regression curves  $c_{\mathbb{S}^{(1)}}(t)$ ,  $c_{\mathbb{S}^{(2)}}(t)$  are shown as red curves. The projected SPs  $\tilde{\mathbb{S}}^{(1)}$ ,  $\tilde{\mathbb{S}}^{(2)}$  are the red points. For the cluster distance calculation, only pairs of projected SPs of the different clusters for which the euclidean distance is less than a certain value (solid black lines) are taken into account. The resulting cluster distance (off-text distance with respect to the average slope of the regression curve at the respective positions) is depicted as cyan line. . . . . 72
- 4.17. **Influence of the separator information on the clustering** – The resulting baselines (blue lines) with and without taking into account the separator information are shown. . . . . 74
- 4.18. **Influence of the number of training samples and of different data augmentation strategies** – The bar height represents the mean F-value. The error bars encode min-max values of the 5 experiments (not the standard deviation). The dashed green line marks the maximum mean value of 0.975 achieved for 350 trainings samples. For a detailed description of the different augmentation strategies: B (no augmentation), S (arbitrary scaling), S+A (S+ affine transformations) and S+A+E (S+A+ elastic transformations), see main text. . . . . 78
- 4.19. **Results for an image of the CSG18 subset of the test set** – The original image (only the main text lines were ground truthed), the baseline image generated by the trained ARU-Net and the baselines detected by the proposed method are shown (from left to right). . . . 85
- 5.1. **Baseline to text line** – This figure illustrates different stages of the proposed baseline to text line methodology. . . . . 88

5.2. <b>Text line extraction results for two snippets of the DIVA-HisDB</b> – Shown are the results of the text line extraction which takes care of the interlinear glosses (see main text) for two snippets of the DIVA-HisDB test set [Sim+16]. A spring factor of 0.001 was used. The visualization was generated by the tool presented in [Alb+17]. Green means correct, blue are false negatives, yellow pixels are assigned to the wrong text line and red pixels are no main body pixels at all. . . . .	92
A.1. <b>Results for an image of the CSG18 subset of the test set with an alternative BLI</b> – Shown is the baseline image generated by the trained ARU-Net and the baselines detected by the proposed method for the image used in Fig. 4.19. In contrast, here, the ARU-Net was trained to detect <b>all</b> text lines, not just the main body text lines. . . .	99
A.2. <b>Results for an image of the Bozen test set</b> – Results for RU-Nets trained on 5, 30 and 350 training samples (left to right) with different data augmentation strategies B, S+A and S+A+E (top to bottom) are shown. . . . .	100
A.3. <b>Results for an image of the Bozen test set</b> – Results for two “degraded” images are shown. The images were arbitrarily curved and rotated. . . . .	101
A.4. <b>Results for images of the cBAD test set [Die+17b]</b> – The images are sampled from the complex track (no layout information was given). . . . .	102
A.5. <b>Results for images of the cBAD test set [Die+17b]</b> – The images are sampled from the complex track (no layout information was given). . . . .	103
A.6. <b>Results for images of the cBAD test set [Die+17b]</b> – The images are sampled from the complex track (no layout information was given). . . . .	104
A.7. <b>Text line extraction results for the CB55 subset</b> – Shown are the results for the text line extraction for a snippet of the CB55 subset of the DIVA-HisDB [Sim+16]. Distance penalty values of (from top to bottom): 0, 0.001, 0.005, 0.1, 0.2, 0.5 were used. The visualization was generated by the evaluation tool presented in [Alb+17]. Green means correct, blue are false negatives, yellow pixels are assigned to the wrong text line and red pixels are no main body pixels at all. . . .	105
A.8. <b>Text line extraction results for the CSG18 subset</b> – Shown are the results for the text line extraction for a snippet of the CB55 subset of the DIVA-HisDB [Sim+16]. Distance penalty values of (from top to bottom): 0, 0.001, 0.005, 0.1, 0.2, 0.5 were used. The visualization was generated by the evaluation tool presented in [Alb+17]. Green means correct, blue are false negatives, yellow pixels are assigned to the wrong text line and red pixels are no main body pixels at all. . . .	106

- A.9. **Text line extraction results for the CSG18 subset** – Shown are the results for the text line extraction for a snippet of the CB55 subset of the DIVA-HisDB [Sim+16]. Distance penalty values of (from top to bottom): 0, 0.001, 0.005, 0.1, 0.2, 0.5 were used. The visualization was generated by the evaluation tool presented in [Alb+17]. Green means correct, blue are false negatives, yellow pixels are assigned to the wrong text line and red pixels are no main body pixels at all. . . . 107
- A.10. **Text line extraction result (GT-BL) for the StAZH dataset** – Shown are the text line extraction results for an image of the StAZH dataset given the ground truth baselines. The trained ATR system yields an CER of 5.80% for the depicted text lines. . . . . 108
- A.11. **Text line extraction result (no GT) for the StAZH dataset** – Shown are the text line extraction results for an image of the StAZH dataset given solely the raw image. The trained ATR system yields an CER of 7.83% for the depicted text lines. The increase in the CER is (mainly) due to the segmentation error connecting marginalia and main body text. . . . . 109
- A.12. **Text line extraction result (GT-TL) for the Bozen dataset** – Shown are the ground truth text lines for an image of the Bozen dataset. The trained ATR system yields an CER of 4.10% for the depicted text lines. . . . . 110
- A.13. **Text line extraction result (no GT) for the Bozen dataset** – Shown are the text line extraction results for an image of the Bozen dataset given solely the raw image. The trained ATR system yields an CER of 4.36% for the depicted text lines. Hence, a comparable performance is achieved for an entirely different TLI. Note that the TLI has to be consistent between training set and test set. For instance, the ATR system of Fig. A.12 yields a CER of 4.87% for this TLI. . . . . 111



# List of Tables

4.1.	<b>Examples for the coverage function</b> – Example values of the coverage function for the normalized polygonal chains shown in Fig. 4.4b with a fixed tolerance value of 20 (as shown in Fig. 4.4d). $\mathcal{G}^{(i)}$ means the normalized version of the $i$ -th GT baseline. . . . .	49
4.2.	<b>Examples for the R-, P-, and F-values</b> – The values for different subsets of the GT and HYP baselines shown in Fig. 4.4a are calculated. A fixed tolerance value of 20 is used for the calculations. . . . .	52
4.3.	<b>Relation between F-value and computational time</b> – Shown are F-values and the time required for their computation for several distance parameters. The experiments were performed for an exemplary hypothesis set for the cBad test set (complex track), see [Die+17a]. The test set is composed of 1010 pages of heterogeneous historical documents. . . . .	54
4.4.	<b>Set of possible interline spacings</b> – The resulting interline spacings for the chosen projection profile diameters $d \in \{64, 128, 256, 512\}$ and fourier coefficient indices $k \in \{3, 4, 5\}$ are shown. . . . .	68
4.5.	<b>Hyperparameters</b> – The architecture and training configuration which were used in this work are described. . . . .	77
4.6.	<b>Results for the Bozen test set</b> – The results in the Base, Curved and Oriented scenario are depicted. The bold numbers are the averages of 5 experiments. The P- and R-values are strongly related to the well-known precision and recall measures, see Sec. 4.2. Finally, the results for a system trained with all degradations are shown. . . . .	81
4.7.	<b>Results for the cBAD test set</b> – The results for different neural network architectures and the workflow without Stage II (for the ARU-Net) are shown. Each architecture is trained 5 times on the cBAD train set. The results are sorted with respect to computational effort. The last two columns indicate whether an architecture is superior to all before mentioned ones in terms of disjunct confidence intervals and the Tukey-Duckworth test. . . . .	82
4.8.	<b>Origing Point (OP) detection results for the ANDAR-TL test set</b> – Results for the dataset of [Mur+15] are shown. #DF means the number of detection failures (no OP detected by the system), #DM means the number of detection misses (detected OP far away from the ground truth OP) and #FP means the number of false positives. . . . .	83

4.9.	<b>Results for the ICDAR2017 Competition on Layout Analysis for Challenging Medieval Manuscripts (Task 2)</b> – The F-values for Task 2 of all participants and the proposed method are shown for the different subsets of the test set. . . . .	84
4.10.	<b>Results for the cBAD test set</b> – The P-, R- and F-values of all participants and of the proposed method for the simple and complex track of the cBAD: ICDAR2017 Competition on Baseline Detection are shown. The bold numbers display the average value for 5 trained ARU-Nets. . . . .	86
5.1.	<b>Results for the ICDAR2017 Competition on Layout Analysis for Challenging Medieval Manuscripts (Task 3)</b> – The line <i>IU</i> and pixel <i>IU</i> of all submissions, of an additional combined method, and of the proposed method for several spring factors are shown for the three subsets of the test set. Finally, the average value is given. The average line <i>IU</i> constitutes the basis for the competition ranking.	91
5.2.	<b>Results for the end-to-end experiments</b> – Shown are the average character error rate and the average bag of words $F_1$ -score for several experiments. GT-TL means that the ATR system was evaluated on the GT text lines, GT-BL means an evaluation on the text lines extracted given the GT baselines, and no GT means that the text lines were extracted given solely the input image. For the Bozen dataset, the results of [San+16] are listed to provide some comparison (comparable to GT-TL). . . . .	94

# List of Algorithms

1.	Minibatch Gradient Descent: $\boldsymbol{\theta} = MGD(\mathcal{E}, \mathfrak{T}', \boldsymbol{\theta}_0, \eta, n)$ . . . . .	28
2.	$\alpha$ - $\beta$ -Swap Energy Minimization: $\Lambda = SWE(E, l, \mathbb{P}, \mathfrak{N})$ . . . . .	33
3.	Polygonal Chain Normalization: $\tilde{\mathcal{P}} = \text{NORM}(\mathcal{P}, d)$ . . . . .	44
4.	Coverage Function: $c = \text{COV}(\mathcal{P}^{(1)}, \mathcal{P}^{(2)}, t)$ . . . . .	49
5.	Greedy P-Alignment Function: $\mathbb{M} = \text{ALIGN}(\mathbb{G}, \mathbb{H})$ . . . . .	52
6.	Pixel Ground Truth Generation: $\boldsymbol{G} = PGTG(I, \tilde{I}_I(I))$ . . . . .	62
7.	Skeleton Calculation (Lantuéjoul's formula): $I^{(s)} = SKE(I^{(b)})$ . . . . .	65
8.	Local Text Orientation of $\boldsymbol{p}$ : $\omega_{\boldsymbol{p}} = LTO(\boldsymbol{p}, \mathfrak{N}, B)$ . . . . .	67
9.	SP Clustering: $\mathcal{P} = SPC(\mathbb{S}, \mathcal{N})$ . . . . .	76





# Symbols

$\mathfrak{I}$ image space .....	5
$\mathfrak{P}_{TL}$ text line space .....	6
$\mathfrak{T}$ test set .....	7
$\mathfrak{T}'$ train set .....	7
$L_{NLL}$ negative log likelihood .....	23
$L_{SNLL}$ spatial negative log-likelihood .....	26
$\Lambda$ labeling .....	31
$\mathfrak{P}_{BL}$ baseline space .....	41
$\lambda(\cdot; \mathbf{p}, \mathbf{q})$ line segment .....	43
$\omega_{\mathcal{P}}$ polygonal chain orientation .....	44
$\mathbf{o}^{(\alpha)}$ orientation vector .....	45
$\omega_{\mathbf{p}}$ local text orientation .....	65
$s_{\mathbf{p}}$ interline distance .....	65



# Acronyms

<b>ATR</b>	automated text recognition	1
<b>KWS</b>	keyword spotting	1
<b>TLI</b>	text line intuition	6
<b>iff</b>	if and only if	6
<b>GT</b>	ground truth	6
<b>TLE</b>	text line extractor	6
<b>HYP</b>	hypothesis	6
<b>i.i.d.</b>	independent and identically distributed	7
<b>AI</b>	artificial intelligence	11
<b>ML</b>	machine learning	11
<b>DL</b>	deep learning	12
<b>ANN</b>	artificial neural network	12
<b>MLP</b>	multilayer perceptron	13
<b>I/O</b>	input/output	14
<b>CNN</b>	convolutional neural network	15
<b>FCN</b>	fully convolutional network	19
<b>MGD</b>	minibatch gradient descent	28
<b>EMA</b>	exponential moving average	29
<b>BLI</b>	baseline intuition	41
<b>BLD</b>	baseline detector	41
<b>NPL</b>	neural pixel labeler	57
<b>SP</b>	superpixel	62
<b>DFT</b>	discrete fourier transform	67



# Bibliography

- [Aba+15] M. Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015.
- [Ahn+17] B. Ahn, J. Ryu, H. I. Koo, and N. I. Cho. “Textline detection in degraded historical document images”. In: *EURASIP Journal on Image and Video Processing* (2017), p. 82.
- [Alb+17] M. Alberti, M. Bouillon, R. Ingold, and M. Liwicki. “Open Evaluation Tool for Layout Analysis of Document Images”. In: *arXiv* (2017).
- [AMO14] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows*. Elsevier, 2014.
- [AS14] N. Arvanitopoulos and S. Süssstrunk. “Seam Carving for Text Line Extraction on Color and Grayscale Historical Manuscripts”. In: *Proc. of the IAPR International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, 2014, pp. 726–731.
- [BCB14] D. Bahdanau, K. Cho, and Y. Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv* (2014).
- [BD62] R. E. Bellman and S. E. Dreyfus. *Applied dynamic programming*. Princeton University Press, 1962.
- [Ben+07] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. “Greedy layer-wise training of deep networks”. In: *Proc. of the Advances in Neural Information Processing Systems (NIPS)*. 2007.
- [BH69] A. E. Bryson and Y. C. Ho. *Applied optimal control: optimization, estimation, and control*. Blaisdell Pub. Co., 1969.
- [BK04] Y. Boykov and V. Kolmogorov. “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.9 (2004), pp. 1124–1137.
- [BL88] D. S. Broomhead and D. Lowe. *Radial basis functions, multi-variable functional interpolation and adaptive networks*. Tech. rep. Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.
- [BLM17] T. Bluche, J. Louradour, and R. Messina. “Scan, attend and read: End-to-end handwritten paragraph recognition with mdlstm attention”. In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2017, pp. 1050–1055.

- [Blu16] T. Bluche. “Joint line segmentation and transcription for end-to-end handwritten paragraph recognition”. In: *Proc. of the Advances in Neural Information Processing Systems (NIPS)*. 2016, pp. 838–846.
- [BM17] T. Bluche and R. Messina. “Gated Convolutional Recurrent Neural Networks for Multilingual Handwriting Recognition”. In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2017, pp. 13–15.
- [BR89] A. Blum and R. L. Rivest. “Training a 3-node neural network is NP-complete”. In: *Prof. of Advances in Neural Information Processing Systems (NIPS)*. 1989, pp. 494–501.
- [Bre17] T. M. Breuel. “High performance text recognition using a hybrid convolutional lstm implementation”. In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2017, pp. 11–16.
- [Bri88] E. O. Brigham. *The fast Fourier transform and its applications*. Vol. 1. prentice Hall Englewood Cliffs, NJ, 1988.
- [Bry61] A. E. Bryson. “A gradient method for optimizing multi-stage allocation processes”. In: *Proc. Harvard Univ. Symposium on digital computers and their applications*. 1961.
- [BVZ01] Y. Boykov, O. Veksler, and R. Zabih. “Fast approximate energy minimization via graph cuts”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.11 (2001), pp. 1222–1239.
- [BVZ98] Y. Boykov, O. Veksler, and R. Zabih. “Markov random fields with efficient approximations”. In: *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 1998, pp. 648–655.
- [Cau47] A. Cauchy. “Méthode générale pour la résolution des systemes d’équations simultanées”. In: *Comp. Rend. Sci. Paris* 25 (1847), pp. 536–538.
- [CCB15] K. Cho, A. Courville, and Y. Bengio. “Describing multimedia content using attention-based encoder-decoder networks”. In: *IEEE Transactions on Multimedia* 17.11 (2015), pp. 1875–1886.
- [Čer85] V. Černý. “Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm”. In: *Journal of Optimization Theory and Applications* 45.1 (1985), pp. 41–51.
- [Cha+16] W. Chan, N. Jaitly, Q. Le, and O. Vinyals. “Listen, attend and spell: A neural network for large vocabulary conversational speech recognition”. In: *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016, pp. 4960–4964.

- [Che+17] K. Chen, M. Seuret, J. Hennebert, and R. Ingold. “Convolutional Neural Networks for Page Segmentation of Historical Document Images”. In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2017, pp. 965–970.
- [CMS12] D. Ciregan, U. Meier, and J. Schmidhuber. “Multi-column deep neural networks for image classification”. In: *Proc. of the IEEE conference on Computer vision and pattern recognition (CVPR)*. IEEE. 2012, pp. 3642–3649.
- [Col+11] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. “Natural language processing (almost) from scratch”. In: *Journal of Machine Learning Research* 12.Aug (2011), pp. 2493–2537.
- [Cor09] T. H. Cormen. *Introduction to algorithms*. MIT press, 2009.
- [CV95] C. Cortes and V. Vapnik. “Support-vector networks”. In: *Machine learning* 20.3 (1995), pp. 273–297.
- [CW12] T. Causer and V. Wallace. “Building A Volunteer Community: Results and Findings from Transcribe Bentham”. In: *Digital Humanities Quarterly* 6.2 (2012), pp. 1–28.
- [Cyb89] G. Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.
- [D+10] G. Dahl, A.-r. Mohamed, G. E. Hinton, et al. “Phone recognition with the mean-covariance restricted Boltzmann machine”. In: *Proc. of the Advances in Neural Information Processing Systems (NIPS)*. 2010, pp. 469–477.
- [Dah+12] G. E. Dahl, D. Yu, L. Deng, and A. Acero. “Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition”. In: *IEEE Transactions on audio, speech, and language processing* 20.1 (2012), pp. 30–42.
- [Dau+14] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. “Identifying and attacking the saddle point problem in high-dimensional non-convex optimization”. In: *Proc. of the Advances in Neural Information Processing Systems (NIPS)*. 2014, pp. 2933–2941.
- [Del34] B. Delaunay. “Sur la sphere vide”. In: *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk* 7.793-800 (1934), pp. 1–2.
- [DHS11] J. Duchi, E. Hazan, and Y. Singer. “Adaptive subgradient methods for online learning and stochastic optimization”. In: *Journal of Machine Learning Research* 12.Jul (2011), pp. 2121–2159.
- [DHS12] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. John Wiley & Sons, 2012.



- [Die+17a] M. Diem, F. Kleber, S. Fiel, B. Gatos, and T. Grüning. “cBAD: IC-DAR2017 Competition on Baseline Detection”. In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2017, pp. 1355–1360.
- [Die+17b] M. Diem, F. Kleber, S. Fiel, T. Grüning, and B. Gatos. *ScriptNet: IC-DAR 2017 Competition on Baseline Detection in Archival Documents (cBAD)*. 2017.
- [DKS13] M. Diem, F. Kleber, and R. Sablatnig. “Text line detection for heterogeneous documents”. In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2013, pp. 743–747.
- [Doz16] T. Dozat. “Incorporating Nesterov Momentum into Adam”. In: 1. 2016, pp. 2013–2016.
- [Efr87] B. Efron. “Better bootstrap confidence intervals”. In: *Journal of the American Statistical Association* 82.397 (1987), pp. 171–185.
- [EGO17] S. Eskenazi, P. Gomez-Krämer, and J.-M. Ogier. “A comprehensive survey of mostly textual document segmentation algorithms since 2008”. In: *Pattern Recognition* 64 (Apr. 2017), pp. 1–14.
- [Fan+15] H. Fang et al. “From captions to visual concepts and back”. In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1473–1482.
- [Far+13] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. “Learning hierarchical features for scene labeling”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), pp. 1915–1929.
- [Gar+12] A. Garz, A. Fischer, R. Sablatnig, and H. Bunke. “Binarization-free text line segmentation for historical documents based on interest point clustering”. In: *Proc. of the 10th IAPR International Workshop on Document Analysis Systems (DAS)*. IEEE, 2012, pp. 95–99.
- [GB10] X. Glorot and Y. Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proc. of the 13th International Conference on Artificial Intelligence and Statistics*. 2010, pp. 249–256.
- [GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.
- [GG16] Y. Gal and Z. Ghahramani. “A theoretically grounded application of dropout in recurrent neural networks”. In: *Proc. of the Advances in Neural Information Processing Systems (NIPS)*. 2016, pp. 1019–1027.
- [Gir+16] R. Girshick, J. Donahue, T. Darrell, and J. Malik. “Region-based convolutional networks for accurate object detection and segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.1 (2016), pp. 142–158.

- [GMH13] A. Graves, A.-R. Mohamed, and G. Hinton. “Speech recognition with deep recurrent neural networks”. In: *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2013, pp. 6645–6649.
- [Goo+13] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. “Maxout networks”. In: *arXiv* (2013).
- [Gra+09] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. “A novel connectionist system for unconstrained handwriting recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.5 (2009), pp. 855–868.
- [Gra08] A. Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2008, p. 124. ISBN: 978-3-642-24797-2.
- [Grü+17] T. Grüning, G. Leifert, T. Strauß, and R. Labahn. “A Robust and Binarization-Free Approach for Text Line Detection in Historical Documents”. In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2017, pp. 236–241.
- [Grü+18a] T. Grüning, R. Labahn, M. Diem, F. Kleber, and S. Fiel. “READ-BAD: A New Dataset and Evaluation Scheme for Baseline Detection in Archival Documents”. In: *Proc. of the IAPR International Workshop on Document Analysis Systems (DAS)*. 2018.
- [Grü+18b] T. Grüning, G. Leifert, T. Strauß, and R. Labahn. “A Two-Stage Method for Text Line Detection in Historical Documents”. In: *arXiv* (Feb. 2018).
- [GS08] A. Graves and J. Schmidhuber. “Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks”. In: *Proc. of the Advances in Neural Information Processing Systems (NIPS)*. 2008, pp. 545–552.
- [GSL11] B. Gatos, N. Stamatopoulos, and G. Louloudis. “ICDAR2009 handwriting segmentation contest”. In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 14. 1. IEEE, 2011, pp. 25–33.
- [Had08] J. Hadamard. *Mémoire sur le problème d’analyse relatif à l’équilibre des plaques élastiques encastrées*. Mémoires présentés par divers savants à l’Académie des sciences de l’Institut de France: Étrait. Imprimerie nationale, 1908.
- [Has95] M. H. Hassoun. *Fundamentals of artificial neural networks*. MIT press, 1995.
- [He+16] K. He, X. Zhang, S. Ren, and J. Sun. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

- [Heb49] D. O. Hebb. *The Organization of Behavior*. Vol. 911. 1. New York: Wiley, 1949, p. 335. ISBN: 0805843000.
- [Hin+12] G. Hinton et al. “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 82–97.
- [Hor91] K. Hornik. “Approximation capabilities of multilayer feedforward networks”. In: *Neural Networks* 4.2 (1991), pp. 251–257.
- [HOT06] G. E. Hinton, S. Osindero, and Y.-W. Teh. “A Fast Learning Algorithm for Deep Belief Nets”. In: *Neural Computation* 18.7 (2006), pp. 1527–1554.
- [HS06] G. Hinton and R. Salakhutdinov. “Reducing the Dimensionality of Data with Neural Networks”. In: *Science* 313.5786 (2006), pp. 504–507.
- [Hsu02] F.-H. Hsu. *Behind Deep Blue: Building the Computer That Defeated the World Chess Champion*. Princeton, NJ, USA: Princeton University Press, 2002. ISBN: 0691090653.
- [ICH12] A. Isaac, R. Clayphan, and B. Haslhofer. “Europeana: Moving to linked open data”. In: *Information Standards Quarterly* 24.2/3 (2012).
- [IG98] H. Ishikawa and D. Geiger. “Segmentation by grouping junctions”. In: *Proc. Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (1998), pp. 1–7.
- [IS15] S. Ioffe and C. Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *Proc. of the International Conference on Machine Learning (ICML)*. 2015, pp. 448–456.
- [J+15] M. Jaderberg, K. Simonyan, A. Zisserman, et al. “Spatial transformer networks”. In: *Proc. of the Advances in Neural Information Processing Systems (NIPS)*. 2015, pp. 2017–2025.
- [Jae02] H. Jaeger. *A tutorial on training recurrent neuronal networks, covering BPPT, RTRL, EKF and the "echo state network" approach*. Tech. rep. GMD Report 159. German National Research Center for Information Technology, 2002.
- [KB15] D. P. Kingma and J. L. Ba. “Adam: a Method for Stochastic Optimization”. In: *Proc. of the International Conference on Learning Representations (ICLR)*. 2015, pp. 1–15.
- [KC10] H. I. Koo and N. I. Cho. “State estimation in a document image and its application in text block identification and text line extraction”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6312 LNCS.PART 2 (2010), pp. 421–434.

- [KC12] H. I. Koo and N. I. Cho. “Text-line extraction in handwritten Chinese documents based on an energy minimization framework.” In: *IEEE Transactions on Image Processing* 21.3 (2012), pp. 1169–75.
- [KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. “Optimization by Simulated Annealing”. In: *Science* 220.4598 (1983), pp. 671–680.
- [KKG16] T. Konidakis, A. L. Kesidis, and B. Gatos. “A segmentation-free word spotting method for historical printed documents”. In: *Pattern Analysis and Applications* 19.4 (2016), pp. 963–976.
- [KL51] S. Kullback and R. A. Leibler. “On information and sufficiency”. In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.
- [Koo16] H. I. Koo. “Text-Line Detection in Camera-Captured Document Images Using the State Estimation of Connected Components”. In: *IEEE Transactions on Image Processing* 25.11 (2016), pp. 5358–5368.
- [KSH12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Proc. of the Advances in Neural Information Processing Systems (NIPS)*. 2012, pp. 1097–1105.
- [KSZ14a] R. Kiros, R. Salakhutdinov, and R. S. Zemel. “Multimodal neural language models”. In: *Proc. of the International Conference on Machine Learning (ICML)*. 2014, pp. 595–603.
- [KSZ14b] R. Kiros, R. Salakhutdinov, and R. S. Zemel. “Unifying visual-semantic embeddings with multimodal neural language models”. In: *arXiv* (2014).
- [Kum+16] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher. “Ask me anything: Dynamic memory networks for natural language processing”. In: *Proc. of the International Conference on Machine Learning (ICML)*. 2016, pp. 1378–1387.
- [LB88] D. Lowe and D. Broomhead. “Multivariable functional interpolation and adaptive networks”. In: *Complex systems* 2.3 (1988), pp. 321–355.
- [LBH15] Y. A. LeCun, Y. Bengio, and G. E. Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444.
- [LeC+89] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. “Back-Propagation Applied to Handwritten Zip Code Recognition”. In: *Neural Computation* 1.4 (1989), pp. 541–551.
- [LeC+90] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. “Handwritten Digit Recognition with a Back-Propagation Network”. In: *Proc. of the Advances in Neural Information Processing Systems (NIPS)*. Ed. by D. S. Touretzky. Morgan Kaufmann, 1990, pp. 396–404.

- [LeC+98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2323.
- [LeC85] Y. LeCun. “Une procédure d’apprentissage pour réseau à seuil asymétrique”. In: *Proceedings of Cognitiva 85, Paris* (1985), pp. 599–604.
- [LeC98] Y. LeCun. *The MNIST database of handwritten digits*. 1998.
- [Lei+16] G. Leifert, T. Strauß, T. Grüning, W. Wustlich, and R. Labahn. “Cells in Multidimensional Recurrent Neural Networks”. In: *Journal on Machine Learning Research* 17.1 (2016), pp. 3313–3349.
- [Lei76] G. W. Leibniz. “Memoir using the chain rule”. In: (1676).
- [LH696] G. F. A. L’Hôpital. *Analyse des infiniment petits, pour l’intelligence des lignes courbes*. Paris: L’Imprimerie Royale, 1696.
- [Li+08] Y. Li, Y. Zheng, D. Doermann, and S. Jaeger. “Script-independent text line segmentation in freestyle handwritten documents”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.8 (2008), pp. 1313–1329.
- [Lin70] S. Linnainmaa. “The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors”. PhD thesis. Univ. Helsinki, 1970.
- [Lin76] S. Linnainmaa. “Taylor expansion of the accumulated rounding error”. In: *BIT Numerical Mathematics* 16.2 (1976), pp. 146–160.
- [LSD15] J. Long, E. Shelhamer, and T. Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proc. of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 3431–3440.
- [M+14] V. Mnih, N. Heess, A. Graves, et al. “Recurrent models of visual attention”. In: *Proc. of the Advances in Neural Information Processing Systems (NIPS)*. 2014, pp. 2204–2212.
- [Mao+14] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille. “Deep captioning with multimodal recurrent neural networks (m-rnn)”. In: *arXiv* (2014).
- [MB01] U.-V. Marti and H. Bunke. “Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system”. In: *Hidden Markov models: applications in computer vision*. World Scientific, 2001, pp. 65–90.
- [MBi06] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006, p. 738. ISBN: 9780387310732.

- [Mik+10] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur. “Recurrent neural network based language model”. In: *Proc. of the Annual Conference of the International Speech Communication Association*. 2010.
- [Mik+11] T. Mikolov, S. Kombrink, L. Burget, J. Černocký, and S. Khudanpur. “Extensions of recurrent neural network language model”. In: *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2011, pp. 5528–5531.
- [Mit97] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997, p. 414. ISBN: 0070428077.
- [MKW17] B. Moysset, C. Kermorvant, and C. Wolf. “Full-Page Text Recognition: Learning Where to Start and When to Stop”. In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2017, pp. 871–876.
- [ML43] L. F. Menabrea and A. A. Lovelace. *Sketch of the Analytical Engine invented by Charles Babbage*. 1843.
- [Moy+15] B. Moysset, C. Kermorvant, C. Wolf, and J. Louradour. “Paragraph text segmentation into lines with Recurrent Neural Networks”. In: *Proc. of the IAPR International Conference on Document Analysis and Recognition, ICDAR*. IEEE, 2015, pp. 456–460.
- [Moy+17] B. Moysset, J. Louradour, C. Kermorvant, and C. Wolf. “Learning text-line localization with shared and local regression neural networks”. In: *Proc. of the IAPR International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, 2017, pp. 1–6.
- [MP43] W. S. McCulloch and W. Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.
- [Mur+15] M. Murdock, S. Reid, B. Hamilton, and J. Reese. “ICDAR 2015 competition on text line detection in historical documents”. In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2015, pp. 1171–1175.
- [MZ12] T. Mikolov and G. Zweig. “Context dependent recurrent neural network language model.” In: *SLT 12* (2012), pp. 234–239.
- [NG09] A. Nicolaou and B. Gatos. “Handwritten text line segmentation by shredding text into its lines”. In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2009, pp. 626–630.
- [NHH15] H. Noh, S. Hong, and B. Han. “Learning deconvolution network for semantic segmentation”. In: *Proc. of the IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1520–1528.

- [NW12] T. M. Nguyen and Q. M. J. Wu. “Gaussian-mixture-model-based spatial neighborhood relationships for pixel labeling problem”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 42.1 (2012), pp. 193–202.
- [OG12] R. H. J. M. Otten and L. P. P. P. van Ginneken. *The annealing algorithm*. Vol. 72. Springer Science & Business Media, 2012.
- [Par85] D. B. Parker. *Learning-Logic*. Tech. rep. TR-47. Center for Comp. Research in Economics and Management Sci., MIT, 1985.
- [Pol64] B. T. Polyak. “Some methods of speeding up the convergence of iteration methods”. In: *USSR Computational Mathematics and Mathematical Physics* 4.5 (1964), pp. 1–17.
- [Pow11] D. Powers. “Evaluation: From Precision, Recall and F-Measure To Roc, Informedness, Markedness & Correlation”. In: *Journal of Machine Learning Technologies* 2.1 (2011), pp. 37–63.
- [PP93] N. R. Pal and S. K. Pal. “A review on image segmentation techniques”. In: *Pattern recognition* 26.9 (1993), pp. 1277–1294.
- [Pra+16] I. Pratikakis, K. Zagoris, J. Puigcerver, A. H. Toselli, and E. Vidal. “ICFHR2016 Handwritten Keyword Spotting Competition (H-KWS 2016)”. In: *Proc. of the IAPR International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, Oct. 2016, pp. 613–618.
- [PS91] J. Park and I. W. Sandberg. “Universal approximation using radial-basis-function networks”. In: *Neural computation* 3.2 (1991), pp. 246–257.
- [PTV14] J. Puigcerver, A. H. Toselli, and E. Vidal. “Word-Graph and Character-Lattice Combination for KWS in Handwritten Documents”. In: *Proc. of the IAPR International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, 2014, pp. 181–186.
- [Pui17] J. Puigcerver. “Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition?” In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2017, pp. 67–72.
- [Rab89] L. R. Rabiner. “A tutorial on hidden Markov models and selected applications in speech recognition”. In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286.
- [Ran+07] M. A. Ranzato, F. Huang, Y. Boureau, and Y. LeCun. “Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition”. In: *Proc. of the Computer Vision and Pattern Recognition Conference (CVPR)*. IEEE, 2007, pp. 1–8.

- [Ren+17] G. Renton, C. Chatelain, S. Adam, C. Kermorvant, and T. Paquet. “Handwritten text line segmentation using Fully Convolutional Network”. In: *Proc. of the International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2017, pp. 5–9.
- [RFB15] O. Ronneberger, P. Fischer, and T. Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Proc. of the Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. 2015, pp. 234–241.
- [RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1”. In: ed. by D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group. Cambridge, MA, USA: MIT Press, 1986. Chap. Learning I, pp. 318–362. ISBN: 0-262-68053-X.
- [RKC14] J. Ryu, H. I. Koo, and N. I. Cho. “Language-independent text-line extraction algorithm for handwritten documents”. In: *IEEE Signal Processing Letters* 21.9 (2014), pp. 1115–1119.
- [Rom+15] V. Romero, J. A. Sanchez, V. Bosch, K. Depuydt, and J. De Does. “Influence of text line segmentation in Handwritten Text Recognition”. In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2015, pp. 536–540.
- [Ros58] F. Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386.
- [Ros61] F. Rosenblatt. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Tech. rep. Cornell Aeronautical Lab Inc Buffalo NY, 1961.
- [Rus+15] M. Rusiñol, D. Aldavert, R. Toledo, and J. Lladós. “Efficient segmentation-free keyword spotting in historical document collections”. In: *Pattern Recognition Letters* 48.2 (2015), pp. 545–555.
- [SAE14] R. Saabni, A. Asi, and J. El-Sana. “Text line extraction for historical document images”. In: *Pattern Recognition Letters* 35.1 (2014), pp. 23–33.
- [Sán+13] J. A. Sánchez, G. Mühlberger, B. Gatos, P. Schofield, K. Depuydt, R. M. Davis, E. Vidal, and J. de Does. “tranScriptorium: a european project on handwritten text recognition”. In: *Proc. of the ACM symposium on Document engineering*. ACM. 2013, pp. 227–228.
- [San+14] J. A. Sanchez, V. Romero, A. H. Toselli, and E. Vidal. “ICFHR2014 Competition on Handwritten Text Recognition on Transcriptorium Datasets (HTRtS)”. In: *Proc. of the IAPR International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, 2014, pp. 785–790.



- [San+16] J. A. Sanchez, V. Romero, A. H. Toselli, and E. Vidal. “ICFHR2016 competition on handwritten text recognition on the READ dataset”. In: *Proc. of the IAPR International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, 2016, pp. 630–635.
- [Sán+16] J. A. Sánchez, V. Romero, A. H. Toselli, and E. Vidal. *READ dataset Bozen*. 2016.
- [SBY17] B. Shi, X. Bai, and C. Yao. “An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.11 (2017), pp. 2298–2304.
- [Sch15] J. Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural networks* 61 (2015), pp. 85–117.
- [Ser11] B. Series. “Studio encoding parameters of digital television for standard 4: 3 and wide-screen 16: 9 aspect ratios”. In: (2011).
- [Ser82] J. Serra. *Image Analysis and Mathematical Morphology*. Vol. 1. 2. 1982, p. 610. ISBN: 0126372403.
- [Seu+17] M. Seuret, M. Alberti, R. Ingold, and M. Liwicki. “PCA-Initialized Deep Neural Networks Applied To Document Image Analysis”. In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2017.
- [SF16] S. Sudholt and G. A. Fink. “PHOCNet : A Deep Convolutional Neural Network for Word Spotting in Handwritten Documents”. In: *Proc. of the IAPR International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, 2016, pp. 1–6.
- [Sil+16] D. Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *Nature* 529.7587 (2016), pp. 484–489.
- [Sil+17] D. Silver et al. “Mastering the game of go without human knowledge”. In: *Nature* 550.7676 (2017), p. 354.
- [Sim+16] F. Simistira, M. Seuret, N. Eichenberger, A. Garz, M. Liwicki, and R. Ingold. “DIVA-HisDB: A precisely annotated large dataset of challenging Medieval manuscripts”. In: *Proc. of the IAPR International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, 2016, pp. 471–476.
- [Sim+17] F. Simistira, M. Bouillon, M. Seuret, M. Würsch, M. Alberti, R. Ingold, and M. Liwicki. “ICDAR2017 Competition on Layout Analysis for Challenging Medieval Manuscripts”. In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2017, pp. 1361–1370.
- [SM17] S. Sonoda and N. Murata. “Neural network with unbounded activation functions is universal approximator”. In: 43.2 (Sept. 2017), pp. 233–268.

- [SMG13] A. M. Saxe, J. L. McClelland, and S. Ganguli. “Exact solutions to the nonlinear dynamics of learning in deep linear neural networks”. In: (2013), pp. 1–9.
- [Sri+14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. “Dropout: A simple way to prevent neural networks from overfitting”. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.
- [SSG09] Z. Shi, S. Setlur, and V. Govindaraju. “A steerable directional local profile technique for extraction of handwritten arabic text lines”. In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2009, pp. 176–180.
- [SSP03] P. Simard, D. Steinkraus, and J. C. Platt. “Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis”. In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. 2003, pp. 958–963.
- [Str+16a] T. Strauß, T. Grüning, G. Leifert, and R. Labahn. “CITlab ARGUS for Keyword Search in Historical Handwritten Documents: Description of CITlab’s System for the ImageCLEF 2016 Handwritten Scanned Document Retrieval Task”. In: *CEUR Workshop Proceedings*. Évora, Portugal, Sept. 2016.
- [Str+16b] T. Strauß, G. Leifert, T. Grüning, and R. Labahn. “Regular expressions for decoding of neural network outputs”. In: *Neural Networks* 79 (2016), pp. 1–11.
- [Str17] T. Strauß. “Decoding the output of neural networks - A discriminative approach”. PhD thesis. Universität Rostock, 2017.
- [Sus15] D. Sussillo. “Random Walks: Training Very Deep Nonlinear Feed-Forward Networks with Smart Initialization”. In: *arXiv* (2015).
- [Sut+13] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. “On the importance of initialization and momentum in deep learning”. In: *Proc. of the International Conference on Machine Learning (ICML)*. 2013, pp. 1139–1147.
- [SVL14] I. Sutskever, O. Vinyals, and Q. V. Le. “Sequence to sequence learning with neural networks”. In: *Proc. of the Advances in Neural Information Processing Systems (NIPS)*. 2014, pp. 3104–3112.
- [Ten+17] C. Tensmeyer, B. Davis, C. Wigington, I. Lee, and B. Barrett. “PageNet: Page Boundary Extraction in Historical Handwritten Documents”. In: *Proc. of the 4th International Workshop on Historical Document Imaging and Processing*. New York, NY, USA: ACM, 2017, pp. 59–64.

- [TH12] T. Tieleman and G. Hinton. *Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude*. COURSERA: Neural Networks for Machine Learning. 2012.
- [Tuk59] J. W. Tukey. “A Quick Compact Two Sample Test To Duckworth’s Specifications”. In: *Technometrics* 1.1 (1959), pp. 31–48.
- [Vin+15] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. “Show and tell: A neural image caption generator”. In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 3156–3164.
- [Vo+17] Q. N. Vo, S. H. Kim, H. J. Yang, and G. Lee. *Binarization of degraded document images based on hierarchical deep supervised network*. 2017.
- [VZ99] O. Veksler and R. Zabih. “Efficient graph-based energy minimization methods in computer vision”. In: (1999).
- [WAH92] J. Weng, N. Ahuja, and T. S. Huang. “Cresceptron: a self-organizing neural network which grows adaptively”. In: *Proc. of the International Joint Conference on Neural Networks (IJCNN)*. Vol. 1. IEEE, 1992, pp. 576–581.
- [Wer81] P. J. Werbos. “Applications of Advances in Nonlinear Sensitivity Analysis”. In: *Proc. of the 10th IFIP Conference*. 1981, pp. 762–770.
- [Wer88] P. J. Werbos. “Generalization of backpropagation with application to a recurrent gas market model”. In: *Neural Networks* 1.4 (1988), pp. 339–356.
- [WH60] B. Widrow and M. Hoff. *Adaptive switching circuits*. Tech. rep. 4. Stanford Univ CA, Stanford Electronics Labs, 1960, pp. 96–104.
- [Wig+17] C. Wigington, S. Stewart, B. Davis, B. Barrett, B. Price, and S. Cohen. “Data Augmentation for Recognition of Handwritten Words and Lines Using a CNN-LSTM Network”. In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2017, pp. 639–645.
- [WT16] F. Wang and D. M. J. Tax. “Survey on the attention based RNN model and its applications in computer vision”. In: *arXiv* (2016).
- [Xu+15] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. “Show, attend and tell: Neural image caption generation with visual attention”. In: *Proc. of the International Conference on Machine Learning (ICML)*. 2015, pp. 2048–2057.
- [Zah+07] A. Zahour, L. Likforman-Sulem, W. Boussalaa, and B. Taconet. “Text Line segmentation of historical Arabic documents”. In: *Proc. of the IAPR International Conference on Document Analysis and Recognition (ICDAR)*. 2-4. IEEE, 2007, pp. 138–142.

- [Zei12] M. D. Zeiler. “ADADELTA: An Adaptive Learning Rate Method”. In: *arXiv* (2012).



# Index

- A-Net, 60
- activation function, 13
- adjacent, 32, 40
- alignment, 50
- artificial intelligence, 11
- ARU-Net, 60
- backpropagation, 27
- baseline, 40
  - detector, 41
  - ground truth, 41
  - hypothesis, 41
  - intuition, 41
  - similarity score, 41, 51
  - space, 41
  - test score, 41
  - test set, 41
  - train set, 41
- bias, 13
- class label, 20
- classes, 20
- classification function, 20
  - spatial, 24
- cluster, 70
- connected, 33
- connectivity function, 66
- convolution
  - discrete, 15
  - matrix, 16
- convolutional neural network, 18
- coordinates, 5
- coverage function, 47
- curvilinearity value, 71
- data term, 32, 68
- deep learning, 12
- depth
  - representative, 16
- disciminative models, 20
- discriminant function, 20
- distribution
  - data, 20
  - posterior, 20
- edge, 32
- error function, 27
- exponential moving average, 29
- F-value, 51
- feature, 11
- feature map, 17
- flatten function, 18
- forward propagation, 14, 18
- fully convolutional network, 19
- generaitve models, 20
- gradient descent, 27
  - batch, 28
  - minibatch, 28
- graph, 33
  - minimal cut, 34
  - weighted, 34
- greedy P-alignment, 51
- ground truth
  - pixel, 57
- height, 16
- hyper parameter, 15, 19
- image
  - binary, 5
  - colored, 5

- gray-scale, 5
- image space, 5
- in-text distance, 45
- input, 14
- intensity, 5
- interline distance, 65
- intersection over union, 90
- kernel, 17
- labeling, 31
  - energy, 31, 69
  - problem, 31
- layer, 14
  - convolutional, 17
  - output, 14
- learning rate, 28
- line segment, 43
- local text orientation, 65
- logit, 13
  - convolutional, 17
  - convolutional function, 17
  - function, 13
- loss index, 22
- machine learning, 11
- max pooling layer, 18
- minibatch, 28
- minimum cut problem, 34
- model parameter, 15, 19
- multilayer perceptron, 13, 14
- negative log likelihood, 23
  - spatial, 26
- neighborhood system, 32
- neural pixel labeler, 57
- off-text distance, 45
- one-hot encoding, 22
- orientation vector, 45
- output, 14
- overfitting, 7, 27
- P-value, 50
- partition, 32
- path, 34
- pixel, 5
- polygonal chain, 40
  - closed, 40
  - orientation, 44
- preceptron, 13
- probabilistic model, 20
- R-value, 49
- receptive field, 18
- residual block, 58
- RMSProp, 29
- rotation matrix, 71
- RU-Net, 59
- seam, 87
  - medial, 89
  - separating, 89
- semi-metric, 32
- similarity score, 7
- simulated annealing, 31
- smoothing term, 32, 69
- Sobel image, 89
- softmax, 14
  - convolutional, 17
- spatial dimension, 16
- spatial t-range, 71
- spring model, 89
- state, 65
- stochastic gradient descent, 28
- stride, 16
- subsampling factor, 18
- superpixel, 62
- supervised learning, 26
- terminal, 34
- test score, 7
- test set, 7, 26
- text line, 6
  - ground truth, 6
  - hypothesis, 6
  - intuition, 6
  - space, 6
- tolerance value, 44
- total baseline energy, 74
- train set, 7, 26

training, 27

unit, 14

unsupervised learning, 29

vertex, 33

weight function, 34

weights, 13

width, 16

Xavier initialization, 30





# Statement of Originality

I hereby confirm that I have written the present thesis by myself, without contributions from any sources other than those cited in the text, the references, and the acknowledgments.

This applies explicitly to all graphics, drawings, maps and images included in the thesis.

Rostock, 27.04.2018

Tobias Grüning